P. 18474/93

1 1993

informatyka

WYDAWNICTWO SIGMA NOT

Cena 25 000 zł

informatyka

nr 1 1993
Styczeń
Rok wydania XXVIII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr inż. Piotr FUGLEWICZ
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
mgr inż. Jan RYŻKO
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu
00-950 Warszawa
ul. Mazowiecka 12
telefon: 27-43-66
telefaks: 19-21-87
teleks: 814877

W numerze:

	Strona
Od sieci neuronowych do neurokomputerów – *Tomasz Berus, Bohdan Macukow*	1
Obiektowe bazy danych – *Waldemar Wieczerzycki*	8
CeBIT'93 – *WK*	12
STAND88 – język współbieżnego programowania w logice – *Grzegorz Czajkowski, Krzysztof Zieliński*	13
Środki łączności w przemysłowach rozproszonych systemach sterowania – *Dariusz Caban*	19
Recenzje	
Inteligentne bazy danych – *Stefan Januszewski, Maria Januszewska*	28

W najbliższych numerach:

- **Jarosław Bochiński charakteryzuje wpływ zastosowania matematyki na rozwój metod i narzędzi tworzenia oprogramowania.**
- **Andrzej Bendig-Wielowiejski, Andrzej Ruciński i Jerzy Sarad prezentują zasady i obszary zastosowań elektronicznych kart bezstykowych, wykorzystywanych jako karty kredytowe, rozliczeniowe, kontrolne itp.**
- **Jerzy Kisielnicki zajmuje się informatyczną strukturą zarządzania, jej potrzebami i tendencjami zmian.**
- **Piotr Fuglewicz omawia języki czwartej generacji i porównuje je z systemami CASE.**
- **Piotr Sienkiewicz przedstawia wykorzystanie systemów ekspertowych w zarządzaniu.**

Warunki prenumeraty na 1993 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych. Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK III 0/Warszawa nr 370015-1573-139-11

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie przy ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 25 000 zł, ulgowa 18 750 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 75 000, półroczna 150 000, roczna 300 000
Ulgowa: kwartalna 56 250, półroczna 112 500, roczna 225 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 505/92

P. 1847/93
POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach

TOMASZ BERUS
BOHDAN MACUKOW
Instytut Matematyki
Politechnika Warszawska

Od sieci neuronowych do neurokomputerów

Mniej więcej rok temu, na tych łamach [2], naszkicowaliśmy drogę powstawania nowego pomysłu, nowego rozwiązania teoretyczno-sprzętowego jakim jest neurokomputer. Poczynając od pierwszych, historycznych modeli neuronu, przez bardzo proste sieci neuronowe aż do przemysłowego modelu Hopfielda, staraliśmy się pokazać pojawiające się problemy oraz sposoby ich rozwiązywania. Obecnie postaramy się nieco szerzej przedstawić kilka zaawansowanych modeli sieci neuronowych, które są dzisiaj wykorzystywane również do celów komercyjnych.

Powróćmy jeszcze raz do genezy powstania neurokomputerów. Zafascynowani potęgą obliczeniową, szybkością działania i możliwościami współczesnych komputerów (nawet mikrokomputerów!), często zapominamy, czym one naprawdę są – jedynie doskonałymi (bo bardzo szybkimi) liczydłami.

Te znakomite urządzenia naprawdę są powolne i nieefektywne. Przecież do wykonania każdej operacji komputer wykorzystuje jedynie mikroskopijną część swoich możliwości, używa jedynie kilku spośród swych elementów – podczas gdy pozostałe są nieaktywne. Przecież szeregowy system pracy zmusza go do wykonywania operacji w ustalonej kolejności, a to często prowadzi do długotrwałych obliczeń.

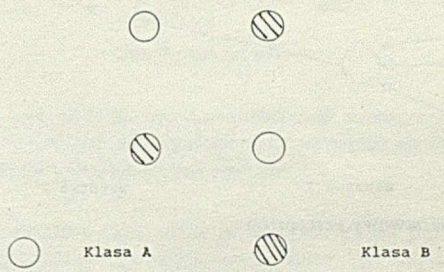
Oczywiście staramy się temu zaradzić. Przez budowę systemów jednostek pracujących równolegle, przez wykorzystywanie procesorów wektorowych itp. możemy zwielokrotnić możliwości obliczeniowe komputera. Jednak wszystkie te usprawnienia nikną wobec potencjalnych możliwości neurokomputerów, w których każdy, niezależnie pracujący neuron, jest jak gdyby niezależnym procesorem, a cała sieć zbudowana z tysięcy (lub milionów) takich procesorów może pracować w pełni równolegle i wykonywać wiele operacji równocześnie. Drugim ułatwieniem oferowanym przez neurokomputery jest odmienne podejście do problemu oprogramowania. Komputer tradycyjny wymaga dokładnych programów, których przygotowanie jest zazwyczaj bardzo czaso- i pracochłonne. Natomiast program pracy neurokomputera zawarty jest w strukturze połączeń między poszczególnymi elementami, a co więcej program ten może być stworzony przy minimalnej ingerencji człowieka – na drodze samouczenia się urządzenia.

Te i wiele innych cech neurokomputerów powodują coraz większe zainteresowanie tą dziedziną oraz stymulują szeroki rozwój badań prowadzonych w wielu ośrodkach naukowych i przemysłowych.

Perceptron wielowarstwowy

W naszym poprzednim artykule [2] w sposób bardzo ogólny omówiliśmy prosty, klasyczny model perceptronu zaproponowanego w 1962 r. przez Franka Rosenblatta. Stosunkowo łatwo można wykazać (co zresztą zrobili w 1969 r. M. Minsky i S. Papert [7]), że możliwości tego typu sieci są bardzo ograniczone. Wiadomo, że klasyczny model perceptronu, składający się z warstwy elementów wejściowych oraz jednego (lub dwóch) elementów w warstwie decyzyjnej, zdolny jest do dokonania podziału zbioru obiektów (obrazów) wejściowych na dwie rozłączne klasy – o ile taki podział jest możliwy.

W przypadku, gdy istnieje rozwiązanie problemu klasyfikacyjnego, reguła ucząca perceptronu zawsze do tego rozwiązania dojdzie. Natomiast klasycznym kontrprzykładem, którego perceptron nie potrafi rozwiązać, jest zagadnienie **XOR** (rys. 1), te dwie klasy nie są liniowo rozdzielne.



Rys. 1. Graficzna interpretacja funkcji XOR
$f(A, B) = \bar{A}B + A\bar{B}$

Dalsze badania wykazały jednak, że te istotne ograniczenia na szczęście nie obowiązują dla struktur wielowarstwowych. Przez dobranie odpowiedniej struktury połączeń oraz wartości wag połączeń możemy otrzymać sieć zdolną do realizacji bardzo złożonych zadań z zakresu klasyfikacji obrazów wejściowych lub tworzenia modeli pamięci skojarzeniowej. Można również konstruować sieci zdolne do realizacji złożonych funkcji logicznych wielu zmiennych i wiele, wiele innych.

Przy budowie struktur wielowarstwowych musimy odpowiedzieć na następujące podstawowe pytanie:
- Ile warstw ma mieć sieć?

Co z prenumeratą INFORMATYKI na 1993 rok?

nr 22013

- Ile elementów ma być w poszczególnych warstwach?
- Jaki będzie algorytm uczenia sieci?

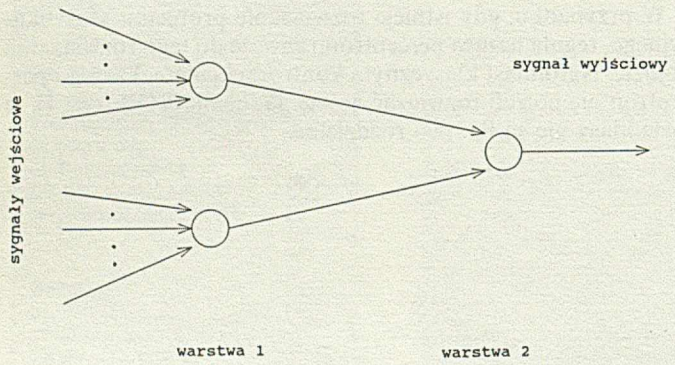
No i oczywiście – na pytanie podstawowe, rzutujące na wszystkie trzy wyżej wymienione:

- Jakie problemy ma budowana sieć rozwiązywać?

Postaramy się pokrótce odpowiedzieć na te pytania.

W literaturze fachowej bardzo często mamy niejednoznaczne podejście do opisu sieci. Wielu autorów, mówiąc o liczbie warstw w sieci, nie precyzuje operacji, jakie wykonują elementy w poszczególnych warstwach. Zazwyczaj elementy pierwszej warstwy nie wykonują żadnej operacji (żadnego sumowania sygnałów czy też nieliniowego ich przekształcania), a jedynie służą do „rozłożenia" sygnału wejściowego na elementy warstwy następnej – w danym przypadku pierwszej warstwy przetwarzającej.

Jak już wspominaliśmy, możliwości jednowarstwowego perceptronu są ograniczone do znajdowania prostej, czy mówiąc ogólniej – hyperpłaszczyzny rozdzielającej dwie klasy. Wyobraźmy sobie sieć o strukturze jak na rysunku 2, złożoną z trzech elementów tego samego typu. Jeżeli element w drugiej warstwie ma wartość progu tak dobraną, aby generował na swym wyjściu sygnał jedynie wtedy, gdy obydwa elementy w pierwszej warstwie są aktywne – układ taki realizuje logiczną operację **AND.** Ponieważ każdy element pierwszej warstwy potrafi określić linię w przestrzeni obrazów – element warstwy drugiej pozwala na dokonanie klasyfikacji opartej na połączeniu tych linii. Gdy warstwa pierwsza ma więcej elementów – na wyjściu sieci otrzymamy jako wynik dowolny zbiór wypukły.



Rys. 2. Dwuwarstwowy perceptron

A co się stanie, gdy dołożymy jeszcze jedną, trzecią warstwę do tej struktury? Elementy w trzeciej warstwie mają na wejściu nie linie dzielące klasy, ale zbiory wypukłe, a więc w rezultacie możemy otrzymać dowolny kształt zbioru określającego daną klasę. Stopień komplikacji złożoność tego zbioru zależy jedynie od liczby elementów w poszczególnych warstwach oraz od doboru wartości progowych. Można udowodnić, że nigdy nie potrzeba więcej niż trzy warstwy w sieci, aby utworzyć dowolnie złożony kształt zbioru określającego klasę.

Oczywiście powyższa analiza dotyczy wykorzystania sieci jako klasyfikatora obrazów, jako urządzenia umożliwiającego podzielenie zbioru obiektów wejściowych na pewną liczbę rozłącznych klas. W przypadku, gdy sieć ma służyć np. jako układ korekcyjny (np. w układach sterowania), jako układ pamięci skojarzeniowej czy układ realizujący złożone operacje logiczne – liczba warstw może być zupełnie inna.

Powracając do analizy omawianego klasyfikatora wielowarstwowego, następnym istotnym problemem jest liczba elementów w poszczególnych warstwach. Liczba neuronów w warstwie wejściowej jest zazwyczaj równa liczbie piksli w obrazach wejściowych. W warstwie wyjściowej można przyjąć, że powinna być to liczba (całkowita) z przedziału $\langle \log_2 M; M \rangle$ gdzie M jest liczbą klas, jakie ma rozróżnić układ. Przykładowo, dla czterech rozłącznych klas możemy skonstruować sieć o dwóch binarnych elementach wyjściowych i poszczególnym klasom będą odpowiadały wektorowe sygnały wyjściowe: [0,0], [0,1], [1,0] i [1,1].

Najtrudniejszym problemem jest określenie niezbędnej liczby elementów w warstwie pośredniej – często nazywanej warstwą ukrytą. W obecnej chwili nie mamy jeszcze precyzyjnie określonych takich warunków, ale zostało udowodnione, że dla N-wymiarowej przestrzeni obrazów wejściowych i K-elementowej warstwy pośredniej – maksymalna liczba rozłącznych klas M jest określona zależnością:

$$M = \sum_{n=0}^{N} \binom{K}{n}, \qquad \text{gdzie dla } K < N \binom{K}{n} = 0.$$

Dla dużych wartości N (dużych w porównaniu z liczbą neuronów w warstwie pośredniej, $N \gg K$) mamy $M = 2^K$. Oznacza to, że dla np. 3-elementowej warstwy pośredniej mamy możliwość dokonania poprawnego podziału zbioru sygnałów wejściowych na maksimum osiem klas, ale ponieważ musi być $N > K$, więc wymiar sygnału wejściowego musi być większy niż 3.

Powyższe rozumowanie można niejako „odwrócić" celem określenia, jaka jest niezbędna liczba K elementów w warstwie pośredniej, aby dokonać prawidłowego podziału na M klas w N-wymiarowej przestrzeni sygnałów wejściowych. W tym celu należy rozwiązać równanie:

$$M = \sum_{n=0}^{N} \binom{K}{n} \qquad \text{dla } K > N$$

a dla $K \leqslant N$ – po prostu $K = \log_2 M$.

Problem wyboru czy zdefiniowania algorytmu uczącego sieci wielowarstwowej jest także zagadnieniem trudnym.

Typowy algorytm uczenia sieci polega na wprowadzaniu na jej wejście wzorcowych sygnałów wejściowych, dla których znamy poprawne sygnały wyjściowe, a następnie takiej przebudowy struktury sieci, aby aktualny sygnał wyjściowy był zgodny z wymaganym. Ponieważ o działaniu zarówno każdego elementu, jak i całej sieci, decydują wagi połączeń międzyelementowych – uczenie polega na zmianie wartości tych wag.

Przyjmuje się, że zmiany wag dotyczą tylko tych elementów sieci, których sygnał wyjściowy był błędny i których sygnał wejściowy był różny od zera (po co zmieniać coś w elementach, które nie były aktywne), a ponadto wielkość zmian jest uzależniona od wielkości błędu, wielkości sygnału wejściowego elementu, a czasem – od jego położenia w strukturze sieci itp.

Stosunkowo łatwo można określić, które z neuronów warstwy wyjściowej mają poprawny sygnał wyjściowy, a które błędny. Natomiast sytuacja jest znacznie trudniejsza w odniesieniu do neuronów warstwy pośredniej. Trzeba także brać pod uwagę, że błędne sygnały wyjściowe elementów warstwy wyjściowej mogły być spowodowane błędnym działaniem (czytaj – błędnymi wagami połączeń) neuronów warstw wcześniejszych. Na podstawie znajomości struktury połączeń, wiedząc skąd dochodziły sygnały do danego elementu, można niejako „przenieść" błędy do warstw wcześniejszych. W ten sposób błędy w sygnałach

wyjściowych neuronów warstwy wyjściowej można, krok po kroku, przenieść aż do warstwy pierwszej – wejściowej. Ponieważ procedura ta przebiega w kierunku przeciwnym w stosunku do kierunku normalnego przekazywania sygnałów w sieci – nazwano ją **algorytmem wstecznej propagacji błędów** (ang. *backpropagation*).

Algorytm wstecznej propagacji błędów

W procesie uczenia do elementów warstwy wejściowej sieci przykładamy wzorcowy sygnał, który po przejściu przez sieć generuje odpowiedni sygnał wyjściowy. Ten aktualny sygnał wyjściowy U^{wy} jest porównywany z sygnałem wzorcowym U^* i jest obliczany globalny sygnał błędu:

$$D = \frac{1}{2} \sum_{j=1}^{M} (U_j^* - U_j^{wy})^2$$

Celem procesu uczenia jest minimalizacja błędu D. Każdą z wartości T_{ji}^{wy}, określającą wagę połączenia między j-tym elementem w warstwie wyjściowej, a i-tym elementem w warstwie pośredniej, „poprawiamy" o wielkość:

$$\Delta T_{ji}^{wy} = -\eta \frac{\alpha D}{\alpha T_{ji}^{wy}} = \eta d_j^{wy} U_i^p$$

gdzie η jest współczynnikiem uczenia,

$d_j^{wy} = -\frac{\alpha D}{\alpha E_j^{wy}}$ jest wielkością lokalnej zmiany błędu dla j-tego elementu,

$E_j^{wy} = \sum_{i=1}^{N} T_{ji}^{wy} U_i^p$ jest ważoną sumą wejściową j-tego elementu warstwy wyjściowej,

$U_i^p = f(E_i^p)$ jest sygnałem wyjściowym i-tego elementu warstwy pośredniej.

Dla elementów warstwy pośredniej otrzymujemy analogiczne zależności:

$$\Delta T_{ji}^p = \eta f'(E_j^p) \sum_{k=1}^{N} d_k^{wy} T_{ki}^{wy}$$

W przypadku, gdy sieć ma więcej niż jedną warstwę pośrednią w analogiczny sposób znajdujemy poprawki ΔT_{ji}^n dla poszczególnych warstw.

Dla bardzo typowej funkcji przenoszenia neuronu:

$$U_j = f(E_j) = [1 + exp(-kE_j)]^{-1}$$

otrzymujemy:

$$\Delta T_{ji}^{wy} = k[U_j^{wy}(1 - U_j^{wy})(U_j^* - U_j^{wy})] U_i^p$$

i odpowiednio:

$$\Delta T_{ji}^p = k[U_j^p(1 - U_j^p) \sum_{k=1}^{M} d_k^{wy} T_{jk}^{wy}] U_i^{we}$$

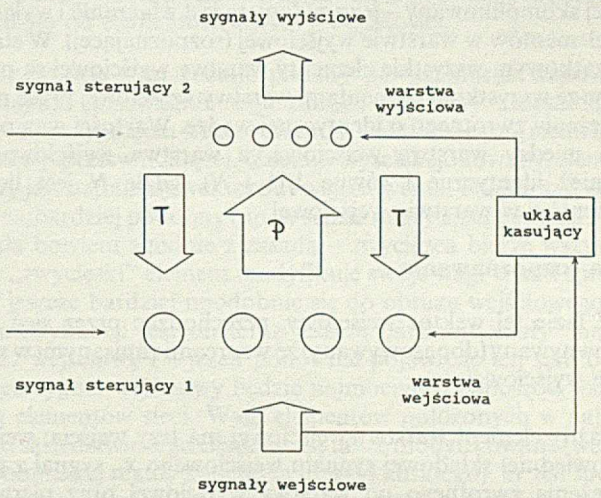
Algorytm uczenia ze wsteczną propagacją błędów jest metodą stosunkowo powolną (szczególnie przy większej liczbie wzorców uczących) – ale jest metodą na tyle skuteczną, że opierając się na jej podstawowych zasadach opracowano (i dalej trwają prace w tym kierunku) dziesiątki jej odmian, które znalazły już zastosowanie w realnych rozwiązaniach, zarówno programowych, jak i sprzętowych.

Model Grossberga — ART

Model ART (*Adaptive Resonance Theory*) został opracowany przez Stephena Grossberga i Gail Carpenter [3], jako układ o strukturze równoległej, będący samoorganizującą się siecią, której działanie jest oparte na wynikach wielu badań biologicznych i neurologicznych.

Architektura sieci ART

Sieć ART jest układem dwuwarstwowym. Warstwa pierwsza jest warstwą wejściowo-porównującą, natomiast warstwa druga – wyjściowo-rozpoznającą (rys. 3).



Rys. 3. Model sieci ART; na rysunku symbolicznie zaznaczono sprzężenia: *P* – w przód, *T* – wsteczne; między elementami występują nie zaznaczone na rysunku sprzężenia typu hamowanie oboczne

Obydwie warstwy są ze sobą powiązane przez połączenia od warstwy wejściowej do wyjściowej oraz do wyjściowej do wejściowej – przez pętle sprzężenia zwrotnego. Ponadto między elementami warstwy wyjściowej istnieje pełen zestaw połączeń typu „hamowanie oboczne".

Dla sieci tej można określić macierz wag połączeń „w przód" od warstwy wejściowej do warstwy wyjściowej oraz macierz połączeń „wstecz", czyli macierz sprzężeń zwrotnych od warstwy wyjściowej do warstwy wejściowej. Są to wagi ulegające zmianie w trakcie procesu samouczenia. Wagi wewnątrz warstwy drugiej (hamowanie oboczne) są stałe nie ulegają zmianom.

Do każdej z warstw docierają również specjalne logiczne sygnały sterujące, które kontrolują przepływ danych przez warstwy w każdym z etapów działania sieci.

Czy prenumerata na bieżący rok została już zamówiona?

W modelu ART poza właściwymi warstwami przetwarzającymi mamy do czynienia z układem kasującym. Poza wyzerowywaniem wyjść elementów warstwy wyjściowej (deaktywacją elementów), dokonuje on porównywania sygnałów wejściowych z wartościami progowymi, co w rezultacie decyduje o zaliczeniu danego sygnału wejściowego do jednej z już istniejących klas, bądź o utworzeniu nowej klasy.

Działanie sieci ART

W procesie uczenia i klasyfikowania sieci ART można wyróżnić następujące cztery podstawowe fazy:

- inicjacji,
- rozpoznawania,
- porównywania,
- przeszukiwania.

Faza inicjacji

W fazie tej istotną rolę odgrywają dwa sygnały sterujące. Pierwszy z nich steruje warstwą wejściową, przełączając elementy sieci między dwoma trybami pracy: przyjmowaniem sygnałów wejściowych oraz porównywaniem. Drugi z sygnałów jest mniej skomplikowany – jego zadaniem jest włączanie i wyłączanie elementów w warstwie wyjściowej (rozpoznającej). W stanie początkowym wszystkie elementy warstwy wyjściowej są połączone ze wszystkimi elementami warstwy wejściowej przez pętle sprzężenia zwrotnego o identycznej wadze. Wartości wag połączeń między warstwą wejściową, a warstwą wyjściową, są również identyczne i równe $1/(1 + N)$, gdzie N jest liczbą elementów w warstwie wejściowej.

Faza rozpoznawania

W fazie tej wektor wejściowy, przechodząc przez sieć, jest porównywany (dopasowywany) ze wzorcami zapisanymi w warstwie wyjściowej.

Każdy element warstwy wejściowej ma trzy wejścia: wejście odpowiedniej składowej sygnału wejściowego x_i, sygnał z pętli sprzężenia zwrotnego od warstwy wyjściowej oraz pierwszy sygnał sterujący. Przepływ informacji przez warstwę wejściową jest regulowany wg zasady „dwa z trzech":

Jeżeli dowolne dwa z trzech wejść są aktywne – sygnał wyjściowy jest równy 1, w wypadku przeciwnym jest równy 0.

Wektor wejściowy jest porównywany ze wzorcem zapisanym w każdym elemencie i zostaje określony wzorzec „najbardziej podobny". Porównanie to następuje przez obliczanie korelacji między sygnałem wejściowym (wektorem wejściowym), a wektorami wag poszczególnych elementów. Wówczas ten element, którego wektor wag jest „najbliższy" wektorowi wejściowemu, ma największy sygnał wejściowy (i wyjściowy). „Zwycięski" element przekazuje teraz zapamiętany wzorzec z powrotem do warstwy wejściowej (porównującej).

Faza porównywania

W tej fazie elementy warstwy wejściowej otrzymują dwa sygnały (dwa wektory): wektor wejściowy (nieznany) oraz wektor wzorcowy z określonego elementu warstwy rozpoznającej. Trzecim sygnałem jest sygnał sterujący, który na tym etapie jest równy 0.

Wektor wejściowy $X = [x_1, \ldots, x_N]$ jest w elementach warstwy wejściowej mnożony (logicznie – piksel po pikslu) przez wektor $T_j = [t_{j1}, t_{j2}, \ldots, t_{jN}]$ pochodzący od aktywnego w tym momencie elementu warstwy wyjściowej. W wyniku tych operacji powstaje nowy wektor wyjściowy warstwy wejściowej, tzw. sygnał podobieństwa, który jest przekazywany do warstwy wyjściowej. Wektor ten jest także przekazywany do układu kasującego, łącznie z aktualnym sygnałem wejściowym X.

Układ kasujący odpowiada za porównanie wektora wejściowego oraz sygnału podobieństwa. Test polega na obliczeniu stosunku liczby jedynek w obu wektorach i porównaniu tego stosunku z określoną wartością progową.

Jeżeli obliczona wartość jest większa od zera, to proces klasyfikacji kończymy, a aktywny element warstwy wyjściowej wskazuje na odpowiedni wzorzec (poprawniej – na klasę wyznaczoną przez ten wzorzec). W przeciwnym wypadku sieć przechodzi w fazę przeszukiwania.

Faza przeszukiwania

W fazie tej poszukujemy innego wzorca (wektora wzorcowego), zapisanego w warstwie wyjściowej, który byłby bardziej zbliżony do wektora wejściowego. Najpierw, aktualnie aktywny element wyjściowy (ten, z którym dotychczas porównywaliśmy wektor wzorcowy) zostaje zablokowany, a jego sygnały wyjściowe – wyzerowane (odpowiedzialny jest za to drugi sygnał sterujący). Operacja ta ma podwójne znaczenie – element ten nie bierze już dalej udziału w analizie podobieństwa, a ponadto binarny sygnał sterujący staje się równy 1, gdyż wszystkie sygnały wyjściowe warstwy wyjściowej (rozpoznającej) są równe 0.

Na wejście podajemy ponownie ten sam wektor wejściowy i przechodzimy do fazy porównywania. Określamy element warstwy wyjściowej, w którym jest zapisany wzorzec najbardziej podobny, ponownie tworzymy odpowiednie połączenia oraz obliczamy sygnał podobieństwa, który jest porównywany z wartością progową. Procedura ta jest powtarzana blokując kolejne elementy warstwy wyjściowej aż do znalezienia tego elementu warstwy wyjściowej, dla którego stosunek liczby jedynek w obu wektorach jest większy od wartości progowej. Gdy element taki nie zostanie odnaleziony, oznacza to, że wektor wejściowy nie należy do żadnej z klas i wówczas albo nie zostaje on rozpoznany, albo można mu przyporządkować jeden z dotychczas nie wykorzystanych elementów warstwy wyjściowej i stworzyć nową klasę.

Wady i zalety modelu ART

Model ART ma wiele znaczących cech, wyróżniających go spośród innych modeli sieci neuronowych. Podstawową zaletą jest tzw. „plastyczna" stabilność. Sieć potrafi bowiem dodawać nowe informacje, modyfikując wiedzę już zapamiętaną, bez jakichkolwiek problemów ze zbieżnością, tzn. z lokalnymi minimami energetycznymi (co np. jest wadą modelu Hopfielda).

Algorytm uczący wymaga tylko jednego przebiegu do stworzenia wewnętrznej reprezentacji zapamiętywanego obiektu. Model ART „radzi sobie" równie dobrze z wzorcami binarnymi, jak i ciągłymi (tzn. o wartościach z określonego przedziału liczb rzeczywistych). Wadą modelu ART jest duża wrażliwość na zakłócenia sygnału wejściowego oraz bardzo skomplikowana, wielofunkcyjna struktura trudna do zaimplementowania w formie „rozproszonej" (pomimo iż była ona wzorcowa na pewnych procesach biologicznych).

Sieci samoorganizujące Kohonena

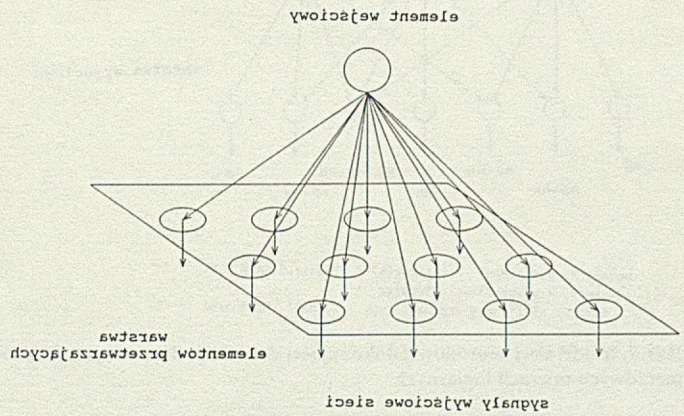
Poprzednio omawialiśmy technikę uczenia sieci, ale uczenia

z nadzorem. Teraz powiemy o metodzie uczenia bez nadzoru – o mapach samoorganizujących Kohonena.

Idea samoorganizacji została zaproponowana w 1973 r. przez von der Malsburga [6]. Jest ona oparta na badaniach dotyczących topologii obszarów korowych mózgu, które wykazały, że określone obszary mózgu, zwłaszcza w korze mózgowej, wykonują wysoce wyspecjalizowane funkcje. Mamy np. dobrze znane i zlokalizowane obszary, odpowiedzialne za mowę, słuch czy ruch. Jak wiadomo, uczenie (czy adaptacja) w układach biologicznych jest procesem chemicznym, który zmienia efektywność połączeń synaptycznych na wejściu do komórki nerwowej. W modelach sieciowych odpowiada to zmianom wag połączeń między elementami sieci.

Aby poprawnie działać, sieci samoorganizujące muszą spełniać dwa podstawowe założenia:
- obrazy wejściowe o określonych wspólnych cechach należą do tej samej klasy,
- sieć jest zdolna do zidentyfikowania tych wspólnych cech w całym zestawie obrazów wejściowych.

Kohonen tworząc swój model [4] wykorzystał te reguły i zaproponował dwuwymiarowy model układu uczącego się.



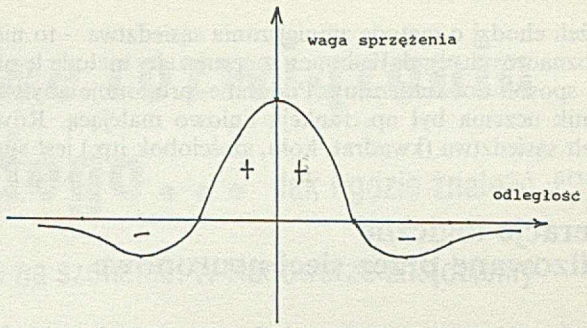
Rys. 4. Model jednowarstwowej sieci samoorganizującej Kohonena; na rysunku zaznaczono połączenia między jednym elementem warstwy wejściowej, a warstwą elementów przetwarzających

Model prostej dwuwymiarowej sieci Kohonena pokazano na rys. 4. Sieć jest zbudowana z warstwy elementów wejściowych, których jedynym zadaniem jest przekazanie sygnałów wejściowych do warstwy następnej – warstwy elementów przetwarzających. Zgodnie z przyjętą umową jest to więc sieć jednowarstwowa. Wyjścia wszystkich elementów warstwy wejściowej są połączone ze wszystkimi elementami sieci. Sieć nie ma specjalnych elementów wyjściowych – każdy element jest elementem wyjściowym.

Wielkość oraz rodzaj połączenia (hamujące czy pobudzające) między elementami warstwy wejściowej i warstwy przetwarzającej zależą od odległości (fizycznej) między elementami.

Uproszczony model wpływu odległości, to tzw. kapelusz meksykański (rys. 5). Neurony położone bliżej elementu wysyłającego sygnały odbierają to oddziaływanie jako silniejsze, a ponadto, przy przekroczeniu określonej progowej wielkości odległości – oddziaływanie to zmienia charakter z pobudzającego na hamujące.

Kohonen opierał się na tych zasadach tworząc zasady połączeń lokalnych i wprowadzając reguły ograniczające obszar zmian adaptacyjnych do pewnego sąsiedztwa.



Rys. 5. Model rozkładu wielkości (i rodzaju) sprzężenia w zależności od odległości źródła sygnału (elementu warstwy wejściowej)

Działanie sieci

W stanie początkowym wszystkie połączenia są małymi liczbami losowymi. Nie jest to jedyna możliwość, można bowiem stosować inne techniki (np. wszystkie wagi są w chwili początkowej stałe i takie same). W fazie uczenia na wejście sieci są podawane obrazy ze zbioru wzorców uczących.

Z każdym elementem warstwy przetwarzającej jest powiązany unikalny wektor. Jest to wektor wag połączeń dochodzących do niego od elementów warstwy wejściowej.

Na wejście sieci zostaje podany pewien obraz wejściowy. Następuje porównanie tego obrazu (możemy utożsamić go z pewnym sygnałem wektorowym) z wektorami wag elementów warstwy przetwarzającej. Zostaje wyselekcjonowany element, który jest najbliższy wektorowi wejściowemu. Najbliższy, oznacza najbardziej podobny (np. w sensie miary Euclidesowej). Sieć działa bowiem zgodnie z zasadą – zwycięzca bierze wszystko. Ten „zwycięski" element modyfikuje swoje wagi w taki sposób, aby jeszcze bardziej upodobnić się do obrazu wejściowego. Po tej modyfikacji jest on teraz jeszcze bardziej „czuły" na ten obraz wejściowy i – jeżeli ponownie pojawi się ten sam obraz – jego sygnał wejściowy będzie najmocniejszy spośród wszystkich elementów sieci. Wagi elementów położonych w najbliższym sąsiedztwie tego elementu są także modyfikowane (według wspomnianej reguły kapelusza meksykańskiego). W ten sposób sieć tworzy obszar, którego elementy reagują silniej niż inne na dany (lub podobny do niego) sygnał wejściowy. Dla różnych sygnałów wejściowych tworzy się układ obszarów, które odpowiadają pewnym klasom w zbiorze sygnałów wejściowych. Sieć określa sama, ile takich klas potrzeba, oraz jakie są między nimi relacje (różnice czy podobieństwa). Ułożenie obszarów w obrębie sieci zależy np. od kolejności prezentowania wzorców.

Zmiana wag połączeń jest proporcjonalna do różnicy między wektorem wejściowym, a wektorem wag sprzężeń. Wielkość tych zmian jest także uzależniona od współczynnika uczenia, który jest malejącą funkcją czasu (czas = liczba prezentacji danego wzorca lub liczba sesji treningowych). Na początku wielkości zmian oraz rozmiary sąsiedztwa są duże. Gdy nastąpi ustabilizowanie się obszarów – elementy w ich ramach są teraz precyzyjnie dopasowywane do wejściowych wzorców treningowych. Typowo, etap precyzyjnego dopasowywania ma 100–1000 razy więcej kroków niż etap zgrubnego tworzenia mapy obszarów.

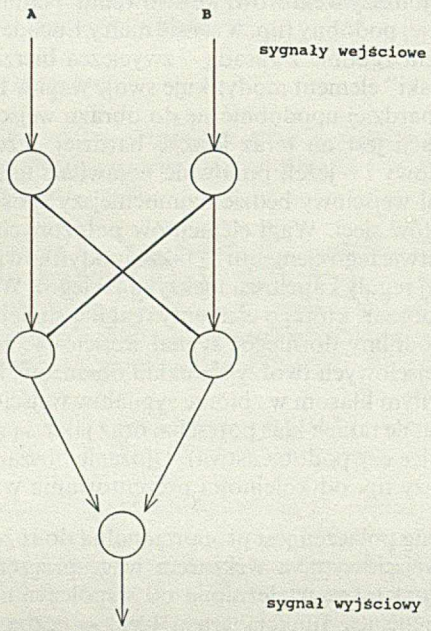
Problem następny, to jak wymodelować ten „kapelusz", jak dobrać wielkości i rodzaj sprzężeń. Od tego bowiem zależy, ile elementów otaczających „zwycięzcę" będzie podlegać zmianom adaptacyjnym w procesie treningowym oraz według jakich zasad. Początkowo każdemu elementowi sieci jest przypisane duże sąsiedztwo (np. cała sieć). W miarę upływu czasu rozmiary sąsiedztwa maleją aż do osiągnięcia przyjętej a priori wielkości granicznej.

Jeżeli chodzi o metodę zmniejszania sąsiedztwa – to nie ma jednoznacznych reguł. Kohonen sugeruje, aby metodę tę określić w sposób doświadczalny. Podobnie, proponuje aby współczynnik uczenia był np. funkcją liniowo malejącą. Również kształt sąsiedztwa (kwadrat, koło, sześciobok itp.) jest sprawą otwartą.

Operacje logiczne realizowane przez sieci neuronowe

Większość prac na temat sieci neuronowych zajmuje się takimi zagadnieniami, jak rozpoznawanie obrazów czy pamięci skojarzeniowe. Trzeba jednak pamiętać, że struktury oparte na zasadach działania sieci neuronowych mają dzisiaj bardzo szerokie zastosowanie.

Jedną z takich ciekawych, a niezbyt jeszcze dokładnie przebadaną, jest możliwość wykorzystania sieci zbudowanych z elementów neuronowych do realizacji funkcji logicznych wielu zmiennych [5]. Przy omawianiu wielowarstwowego perceptronu użyliśmy funkcji XOR jako przykładu operacji nie dającej się zrealizować w jednowarstwowym perceptronie. Tę prostą funkcję logiczną można zrealizować za pomocą sieci neuronowej na wiele różnych sposobów. Jedna z najprostszych struktur pokazana jest na rys. 6. Połączenia pobudzające są oznaczone strzałką i mają wagę równą $+1$, natomiast połączenia bez strzałek są połączeniami hamującymi i mają wagę równą -1. Wartość progu we wszystkich elementach jest taka sama i równa zero. Sygnały wejściowe mogą przyjmować wartości 0 lub 1.



Rys. 6. Prosty model sieci neuronowej realizującej operację XOR; linie zakończone strzałkami reprezentują połączenia o wadze $+1$, natomiast linie bez strzałek – połączenia o wadze -1

Jest to przykład sieci zdolnej do realizacji jednej, konkretnej funkcji logicznej, ale z tego typu elementów można zbudować sieci, które będą zdolne do realizacji bardziej złożonych operacji, np. do realizacji wszystkich istniejących funkcji dwóch zmiennych opisanych zależnością

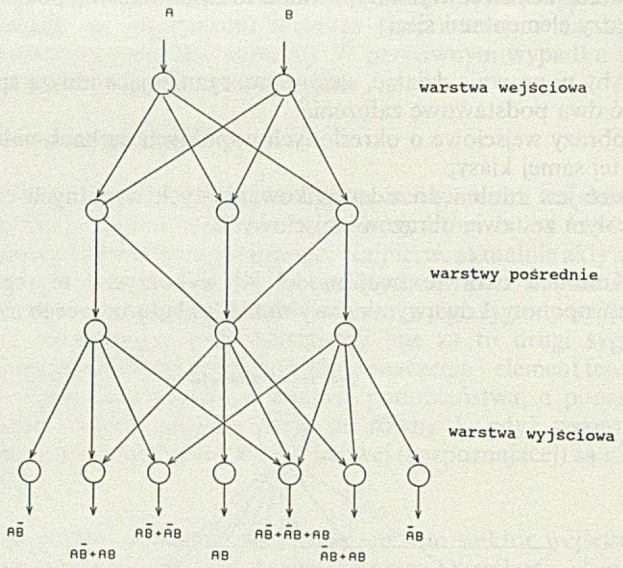
$$f(A,B) = \alpha_0 \bar{A}\bar{B} + \alpha_1 A\bar{B} + \alpha_2 \bar{A}B + \alpha_3 AB$$

Jest to tzw. postać kanoniczna zmiennych logicznych A i B. Przyjmując odpowiednie wartości współczynników α_i (zero lub jeden), otrzymujemy dowolną z funkcji tych zmiennych. Przykładowo, przyjmując $\alpha_1 = \alpha_2 = 1$ i $\alpha_0 = \alpha_3 = 0$, otrzymujemy:

$$f(A,B) = A\bar{B} + \bar{A}B$$

czyli funkcję XOR. Metodę tę można zresztą uogólnić na zagadnienie dowolnej funkcji logicznej n – zmiennych [5].

Na rys. 7 pokazano sieć o wejściach binarnych (0 lub 1), której zasady połączeń i zasady działania są takie same, jak sieci z rys. 6. Każde z wyjść sieci realizuje inną funkcję logiczną dwóch zmiennych. Jak wiadomo, dla dwóch zmiennych mamy ogółem 16 różnych funkcji. Sieć z rys. 7 realizuje siedem spośród tych 16, a mianowicie te, w których różny od zera sygnał wejściowy generuje różny od zera sygnał wyjściowy.




```
(P3)
square(N, N_squared) :- integer(N) | N_squared is N * N.
square(N, N_squared) :- otherwise |
  display_nl({"Wrong argument type : ", N}), N_squared := -1.
```

Rys. 7. Model sieci neuronowej dokonującej siedmiu podstawowych dwuelementowych operacji logicznych

Co możemy uzyskać wykorzystując warstwowe sieci neuronowe do realizacji operacji logicznych?

Sieci realizujące funkcje logiczne mogą być zbudowane z takich samych elementów przetwarzających, jak układy rozpoznające lub pamięci skojarzeniowe [1]. Co więcej, ponieważ sieci te można uczyć wg tych samych reguł, to jednorodność ta pozwoli na znacznie większą elastyczność sieci, ułatwi ewentualne zmiany struktury całego modelu itp. Z tych samych elementów składowych można będzie budować układy realizujące nawet bardzo złożone operacje logiczne na niepełnych lub przekłamanych danych wejściowych. Próby takie są już zresztą robione, a ich wyniki były bardzo obiecujące [1].

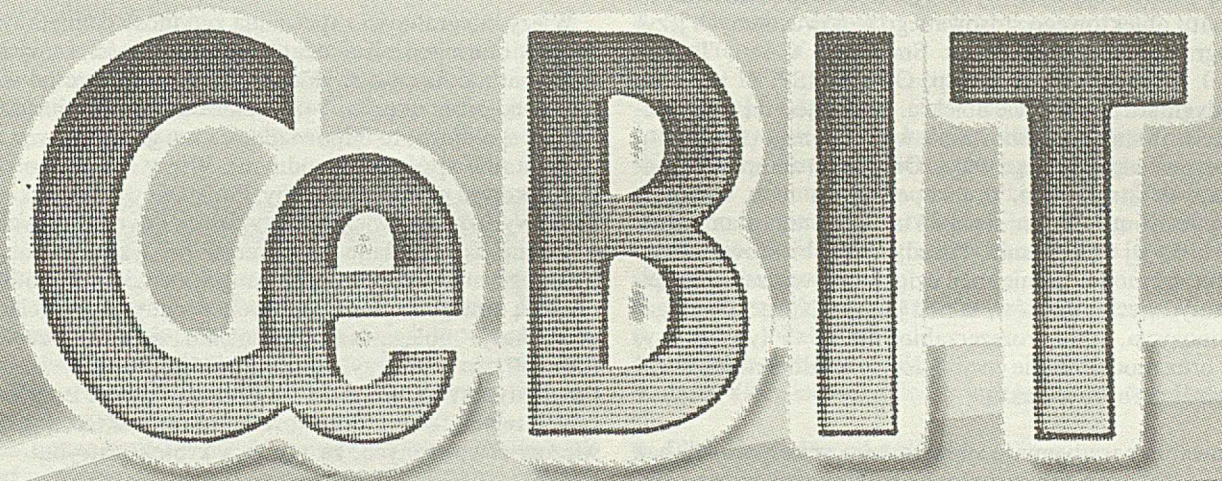
Rosnące zainteresowanie neurokomputerami pozwala przypuszczać, że ta dziedzina badań będzie się nadal szybko rozwijać. Wydaje się, że bliski jest już dzień, gdy zostanie zbudowany prawdziwy neurokomputer, gdy idea przetwarzania równoległego zostanie zrealizowana. Postęp technologiczny jest ogromny. Komputery mamy coraz doskonalsze i coraz szybsze. Ich pamięci i możliwości są coraz większe, a wszystko to mieści się w coraz mniejszym i wygodniejszym „opakowaniu". Dalsze badania, jakie są prawadzone nad udoskonaleniem metod programowania neurokomputerów, to próby skojarzenia metod programowania równoległego z sieciami neuronowymi, natomiast kierunek „sprzętowy" reprezentują dziś próby wykorzystania światła i metod przetwarzania optycznego – czyli neurokomputery optyczne.

dokończenie na s. 18

Optymalne całościowe rozwiązania systemowe dla każdego...

Jak i gdzie znaleźć rozwiązanie trudnego problemu? Nie traćmy czasu na szukanie! W Hanowerze znajdziemy nieograniczone możliwości wyboru. 6000 wystawców z ponad 45 krajów prezentuje całość światowej oferty techniki informatycznej i telekomunikacyjnej. Pierwszy publiczny pokaz najnowszych rozwiązań. Pełna światowa oferta dostępna dla każdego! Zadowoli wszystkich!

CeBIT Hanower 24.– 31. 03. 1993



Światowe Centrum • Biuro • Informacja • Telekomunikacja

Bliższe informacje: MGR ZDZISŁAW A. GEPPERT, PRZEDSTAWICIEL W POLSCE
ul. Mokotowska 49, 00-512 Warszawa
Tel.: (02) 66 00-505, 66 00-525, Telex: 814241 mex pl, Fax: (022) 29 85 59

DEUTSCHE MESSE AG, HANNOVER/GERMANY

WALDEMAR WIECZERZYCKI
Francusko-Polska Wyższa Szkoła
Nowych Technik Komunikacyjno-Informatycznych
Poznań

Obiektowe bazy danych

Teoretyczną podstawą zdecydowanej większości dostępnych obecnie komercyjnych systemów baz danych: Oracle, Ingres, Sybase, Progress, Isis, Superfile, Informix-ESPL/C, dBase IV i Rbase jest relacyjny model danych. Jego podstawowymi zaletami są: prostota struktur danych zredukowanych do relacji, nieproceduralne, deklaratywne języki manipulowania danymi, niezależność fizyczna i logiczna danych oraz optymalizacja dostępu do danych. Zalety te spowodowały, że relacyjne bazy danych znalazły zastosowania w typowych aplikacjach bankowych, magazynowych, biurowych, ewidencji personalnej itp. Specyfiką tych zastosowań są proste struktury danych i związki między nimi, które w łatwy sposób mogą być odwzorowywane w strukturach danych modelu relacyjnego.

Pojawienie się tanich, bardzo efektywnych graficznych stacji roboczych spowodowało zainteresowanie nowymi zastosowaniami systemów baz danych, przede wszystkim w systemach komputerowego wspomagania projektowania i zarządzania, komputerowego wspomagania inżynierii oprogramowania i w systemach informacji biurowej. Zastosowania te wymagają bardziej złożonych struktur danych, umożliwiających reprezentowanie schematów, rysunków, obrazów, dźwięków itp. oraz nowych typów związków między nimi. Ponieważ relacyjny model danych okazał się dla tych zastosowań niewystarczający, rozpoczęto poszukiwania nowych modeli danych. Najbardziej obiecującą propozycją było wykorzystanie w modelu danych paradygmatu obiektowego stosowanego w nowoczesnych językach programowania, takich jak Smalltalk, C++, Turbo-Pascal 6.0, Objective-C, Oak-Lisp, Object-LISP. U podstaw tego paradygmatu leży pojęcie obiektu. Obiekt jest reprezentantem (modelem) stanu i zachowania wyróżnionego fragmentu świata rzeczywistego zwanego encją. Obiekt jest zbiorem zmiennych, zwanych atrybutami, oraz operacji na nich, zwanych metodami. Atrybuty obiektu służą do modelowania stanu encji, a metody – do jej zachowania. Paradygmat obiektowy umożliwia elastyczne modelowanie encji dzięki wprowadzeniu związków semantycznych obiektów, takich jak specjalizacja, agregacja i kompozycja, łatwą rozszerzalność o nowe typy (klasy) obiektów oraz modelowanie encji o dowolnej złożoności dzięki rekurencyjnej definicji obiektu.

Zastosowanie paradygmatu obiektowego w konstrukcji baz danych zaowocowało tak zwanymi obiektowymi bazami danych. Najbardziej znanymi eksperymentalnymi obiektowymi bazami danych są: Orion, 02, GemStone, Ontos, ObjectStore, Exodus, Cactis, Vision, Iris i VBase [3, 5, 6, 7, 8, 10].

Jak ogólnie wiadomo, model relacyjny baz danych charakteryzuje się trzema podstawowymi składowymi: relacyjnymi strukturami danych, dostępnością operatorów algebry relacyjnej, umożliwiających tworzenie, przeszukiwanie i modyfikację danych, oraz więzami integralności, jawnie lub niejawnie określającymi prawidłowe wartości danych [9].

Struktury danych modelu relacyjnego opierają się na pięciu pojęciach:
- dziedziny, będącej zbiorem dopuszczalnych wartości danych,
- relacji, będącej podzbiorem produktu kartezjańskiego wybranych dziedzin, powszechnie przedstawianej w postaci dwuwymiarowej tablicy,
- krotki (wiersza tablicy),
- atrybutu (nazwy kolumny tablicy),
- klucza (identyfikującego wiersz tablicy).

Każda zapisana w relacji wartość jest niezależną daną prostą (liczbą, łańcuchem znaków lub datą), uporządkowanie atrybutów oraz krotek jest nieistotne, a każda krotka relacji musi być różna od pozostałych.

Operatory algebry relacji działają na jednej lub wielu relacjach, a ich wynikiem są inne relacje. Wyróżnia się następujące operatory:
- **selekcji,** umożliwiający wybór krotek relacji spełniających określone warunki,
- **projekcji,** umożliwiający okrojenie relacji do wybranych atrybutów,
- **połączenia,** umożliwiający łączenie krotek wielu relacji,
- **klasyczne operatory teorii mnogości,** przykładowo suma mnogościowa, produkt kartezjański itp.

Więzy integralności zapewniają zgodność pamiętanych w relacjach informacji z fragmentem rzeczywistości, który relacje te reprezentują. Są one zwykle predykatami, które powinny być spełnione przez określony podzbiór danych relacji. Jeśli predykat jest spełniony, to odpowiadająca mu grupa danych znajduje się **w stanie spójnym** (zgodnym). Więzy integralności można klasyfikować na różne sposoby. Podstawowa klasyfikacja polega na wyróżnieniu więzów statycznych i dynamicznych. Więzy statyczne są wyrażane za pomocą predykatów dotyczących bieżącego stanu relacji bazy danych, a więzy dynamiczne za pomocą predykatów dotyczących zmiany stanu relacji. Inna klasyfikacja polega na wyróżnieniu więzów wewnętrznych i zewnętrznych. W przypadku więzów wewnętrznych odpowiednie predykaty są wyrażane za pomocą krotek i atrybutów tej samej relacji, dla której zostały określone, a w przypadku więzów zewnętrznych, za pomocą krotek i atrybutów innych relacji.

Model relacyjny bazy danych ma wiele zalet, które spowodowały, że wyparł on modele stosowane wcześniej: hierarchiczny i sieciowy. Podstawową jego zaletą jest prostota struktur danych, zredukowana do relacji. W strukturach danych modelu relacyjnego nie wprowadza się zatem żadnych pojęć poziomu fizycznego. Zbiory encji i powiązania między nimi są reprezentowane, niezależnie od ich typu, zawsze w ten sam sposób – w postaci dwuwymiarowych tablic. Prostota struktur danych sprawia, że może z nich korzystać bardzo wielu użytkowników,

niekoniecznie będących informatykami. Drugą zaletą jest wyposażenie systemów relacyjnych baz danych w nieproceduralne, deklaratywne języki manipulowania danymi. W językach tych jedynie określa się które informacje mają być wyszukane w bazie danych, natomiast nie definiuje się sposobu ich wyszukania. Stosowanie języków deklaratywnych przyspiesza i ułatwia przygotowanie programów użytkowych, w szczególności osobom nie będącym informatykami. Trzecią zaletą jest niezależność fizyczna i logiczna danych, wynikająca z faktu, że model relacyjny nie zawiera elementów opisu organizacji fizycznej. Definiowanie struktur fizycznych relacji i ścieżek dostępu realizuje się całkowicie niezależnie od definiowania schematu logicznego relacji. Czwartą zaletą, będącą konsekwencją dwóch poprzednich, jest optymalizacja dostępu do bazy danych. Osiągnięcie logicznej i fizycznej niezależności danych oraz korzystanie z języka deklaratywnego daje możliwość określania najlepszej strategii dostępu do danych. Strategia ta jest określana w sposób automatyczny przez system zarządzania bazą danych. Ostatnią zaletą jest oparcie modelu relacyjnego na szczególnie mocnym i precyzyjnym fundamencie teoretycznym.

Zalety relacyjnych baz danych spowodowały, że znalazły one zastosowania w typowych aplikacjach bankowych, biurowych, magazynowych, ewidencji personalnej itp. Specyfiką tych zastosowań są proste struktury danych i związki między nimi, które w łatwy sposób mogą być odwzorowywane w struktury danych modelu relacyjnego. Nowe zastosowania systemów baz danych, przede wszystkim w systemach komputerowego wspomagania projektowania, komputerowego wspomagania inżynierii oprogramowania i w systemach informacji biurowej, charakteryzują się znacznie bardziej złożonymi strukturami danych oraz nowymi typami związków między nimi. Nowa klasa zastosowań stawia przed modelem danych systemu bazy danych pięć podstawowych wymagań:

- Możliwość modelowania złożonych obiektów świata rzeczywistego, z odwzorowaniem ich wielopoziomowej struktury. Wymaganie to wiąże się z wprowadzeniem nowych typów danych, umożliwiających tworzenie uporządkowanych i nieuporządkowanych kolekcji jednorodnych i niejednorodnych danych, z jednej strony, i wprowadzeniem danych typu wskazującego na inne dane, z drugiej strony.

- Rozszerzenie semantyki modelu danych o koncepcje pozwalające na reprezentowanie związków występujących między obiektami świata rzeczywistego, takich jak generalizacja, specjalizacja, agregacja, zawieranie się obiektów oraz wersje obiektów.

- Łatwa rozszerzalność o nowe typy danych, definiowane od podstaw lub na drodze specjalizacji typów istniejących.

- Integracja języka programowania, w którym są pisane aplikacje, z językiem manipulacji danymi w bazie danych. Wymaganie to wynika z niedopasowania tych dwóch typów języków w klasycznych bazach danych. Niedopasowanie to ma dwa aspekty. Pierwszym jest różnica między paradygmatem deklaratywnego języka bazy danych, takiego jak np. SQL, i paradygmatem imperatywnego języka programowania, takiego jak np. C. Drugim aspektem tego niedopasowania są różne systemy typów danych. Klasyczne języki programowania nie pozwalają na bezpośrednie odwzorowywanie struktur danych występujących w bazach danych, takich jak: relacje, krotki i atrybuty. Powoduje to potrzebę dwustronnych transformacji między tymi dwoma systemami typów danych oraz uniemożliwia kontrolę poprawności typów danych w aplikacji jako całości.

- Uzyskanie kompletności obliczeniowej języka manipulacji danymi, czyli możliwości wyrażania w nim dowolnych funkcji matematycznych, logicznych i relacyjnych (klasyczne języki manipulacji danymi nie są kompletne obliczeniowo).

Ponieważ relacyjny model danych okazał się w nowych zastosowaniach niewystarczający, rozpoczęto poszukiwania nowych modeli danych, które miałyby wyżej wymienione cechy. Najbardziej interesującą propozycją było wykorzystanie w modelu danych paradygmatu obiektowego. Obejmuje on większość z wymienionych koncepcji semantycznego modelowania danych oraz, dzięki rekurencyjnej definicji obiektu, umożliwia modelowanie obiektów o dowolnej złożoności. Ponadto, dziedziczenie klas i hermetyczność obiektów zapewniają łatwe rozszerzanie systemu bazy danych o nowe typy danych. Również w dziedzinie języków programowania były czynione wysiłki zmierzające do wzbogacenia ich paradygmatu o cechy systemów baz danych. Uwieńczeniem tych dwustronnych, zbieżnych działań są **obiektowe bazy danych,** będące zintegrowanymi systemami programowania obiektowego i baz danych.

W celu zintegrowania języka obiektowego z systemem bazy danych należy rozwiązać dwa podstawowe problemy. Pierwszym z nich jest wzbogacenie semantyki języka obiektowego o nowe cechy:
- charakterystyczne dla języków baz danych,
- związane z wymaganiami nowych zastosowań systemów baz danych.

Drugim problemem jest zarządzanie obiektami w pamięci operacyjnej i zewnętrznej. Programista aplikacji nie powinien być świadomy systemu zarządzania bazą danych, tzn. powinien móc manipulować obiektami w swoim programie tak, jakby znajdowały się one w nieograniczonej pamięci wirtualnej. System bazy danych musi być wyposażony w takie mechanizmy, całkowicie niewidoczne dla użytkownika, jak: fizyczne grupowanie danych (ang. *clustering*), zarządzanie indeksami, buforowanie operacji we-wy, selekcję ścieżek dostępu oraz optymalizację zapytań. Ze względów efektywnościowych istotna jest troskliwa implementacja tych mechanizmów, gdy operacje we-wy pochłaniają znacznie więcej czasu niż przetwarzanie.

Wśród cech pochodzących z klasycznych języków baz danych wyróżniamy:
- trwałość obiektów,
- współbieżność wykonywania transakcji,
- ochronę spójności danych,
- odtwarzanie danych,
- zapytania *ad hoc*.

Trwałość obiektów (ang. *object persistency*) umożliwia przechowywanie między kolejnymi sesjami użytkownika wszystkich lub wyróżnionych obiektów bazy danych. Pojęcie trwałości można rozumieć jako czas, przez który dane istnieją i są dostępne za pomocą operatorów języka bazy danych. Dane te można podzielić według rosnącego stopnia trwałości na następujące kategorie:
- wyniki przejściowe wartościowych wyrażeń,
- zmienne lokalne metod,
- zmienne globalne aplikacji,
- dane, które istnieją między wykonaniami określonej aplikacji,
- dane niezależne od aplikacji.

Dodanie trwałości do języka obiektowego powinno być realizowane z uwzględnieniem następujących dwóch podstawowych reguł:

- niezależności trwałości danych od sposobu, w jaki manipuluje się nimi w aplikacji i na odwrót, niezależności treści aplikacji od stopnia trwałości danych,

- ortogonalności trwałości i typu danych: wszystkie dane, niezależnie od typu, mogą mieć nadany dowolny stopień trwałości.

Współbieżność wykonywania transakcji jest rozumiana klasycznie jako możliwość przeplatania się operacji zbioru transakcji, w ogólności inicjowanych przez różnych użytkowników, ale tak, aby każdy z nich miał wrażenie wyłączności dostępu do bazy danych. Operacje transakcji nie mogą być przeplatane w sposób dowolny, lecz w taki, by stan bazy danych po zakończeniu wykonywania zbioru transakcji był stanem spójnym. Ze współbieżnością wiąże się ściśle pojęcie **atomowości transakcji,** która polega na uniemożliwieniu częściowej realizacji transakcji – transakcja musi być wykonana całkowicie lub wcale.

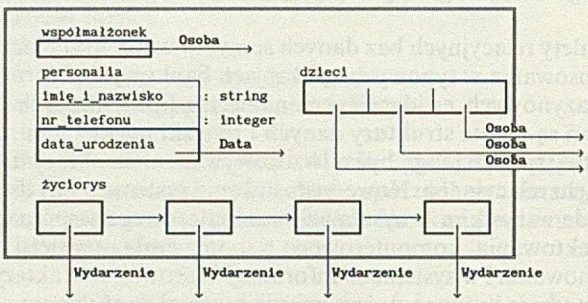
Ochrona spójności danych jest wymagana ze względu na współbieżne wykonywanie transakcji, które realizują dostęp do tych samych danych. W tej sytuacji, aby nie dopuścić do naruszenia spójności danych, system zarządzania bazą danych musi być wyposażony w mechanizmy izolujące wykonywanie każdej transakcji od wpływów innych nie zakończonych transakcji.

Odtwarzanie (ang. *recovery*) jest formą częściowego zabezpieczenia przed awariami sprzętowymi systemu komputerowego oraz niewłaściwymi transakcjami. Funkcja ta umożliwia odtworzenie pewnego spójnego stanu bazy danych z przeszłości, w szczególności tuż przed awarią.

Zapytanie ad hoc jest poprawnym semantycznie i syntaktycznie zapytaniem wyrażonym w języku zapytań oferowanym przez system bazy danych. Zapytanie jest formą programu użytkowego, zapisanego w specjalnym języku wysokiego poziomu. W obiektowych bazach danych takiemu językowi stawia się trzy podstawowe wymagania. Po pierwsze, jego poziom powinien być na tyle wysoki, by można w nim precyzyjnie wyrazić nietrywialne zapytanie za pomocą kilku słów lub przesunięć myszki. Wymaga to możliwie dużej deklaratywności języka. Podobnie jak w klasycznych systemach baz danych, w zapytaniach kierowanych do obiektowej bazy danych nacisk jest położony na to, co należy zrobić, a nie, jak to zrobić. Po drugie, język zapytań powinien być efektywny. Przyjęta składnia języka sama w sobie powinna zmierzać do optymalizacji formułowanych w nim zapytań. Trzecim wymogiem jest niezależność języka zapytań od konkretnej aplikacji.

Wśród cech związanych z wymaganiami nowych zastosowań baz danych wyróżniamy:
- obiekty kompleksowe,
- tożsamość obiektów,
- wersje obiektów.

Obiektem kompleksowym (ang. *complex object*) jest nazywany obiekt zawierający co najmniej jeden atrybut, którego wartością jest wskazanie na inny obiekt. Jeżeli wskazany obiekt nie jest powiązany semantycznie z obiektem wskazującym, to wskazanie nazywamy **wskazaniem słabym** (ang. *weak reference*). Przykładem obiektu kompleksowego zawierającego wskazania słabe jest obiekt *samochód_Waldka,* będący wystąpieniem klasy *Samochód* o atrybutach: *właściciel, marka, rok_produkcji, silnik, nadwozie, podwozie* itp. Atrybut *właściciel* wskazuje na wystąpienie *Waldek* klasy *Osoba,* o atrybutach *personalia, współmałżonek, dzieci* i *życiorys* (rys. 1). Wskazanie to jest wskazaniem słabym, bowiem właściciel nie jest powiązany semantycznie ze swoim samochodem. Obiekt wskazywany może być również obiektem kompleksowym. W powyższym przykładzie, atrybut *współmałżonek* obiektu *Waldek* wskazuje na inne wystąpienie klasy *Osoba – Bożena.* Wskazania między obiektami, będącymi elementami obiektu kompleksowego, tworzą graf skierowany, w którym są dopuszczalne cykle. Zauważmy bowiem, że obiekt *Bożena,* będący wystąpieniem tej samej klasy co *Waldek,* również zawiera atrybut *współmałżonek,* w tym przypadku wskazujący zwrotnie na *Waldka.* Obiekty wskazywane przez obiekt kompleksowy mogą być grupowane za pomocą różnych konstruktorów, na przykład zbioru, krotki lub listy. Konstruktory są ortogonalne do klas wiązanych obiektów. Jedynie w przypadku zbiorów jest wymagana jednorodność obiektów (grupowane obiekty muszą być wystąpieniami tej samej klasy). Warto zauważyć, że warunek ten nie jest spełniony w klasycznych bazach danych, na przykład w relacyjnym modelu danych relacja składa się wyłącznie z krotek, a krotki zawierają wyłącznie wartości proste. Przykładami konstruktorów są trzy kolejne atrybuty wystąpień klasy *Osoba: personalia, dzieci* i *życiorys.* Atrybut *personalia* jest krotką grupującą *imię_i_nazwisko, nr_telefonu* i *datę_urodzenia.* Wartościami dwóch pierwszych atrybutów są odpowiednio łańcuch znaków oraz liczba, a wartością trzeciego atrybutu jest wskazanie na wystąpienie klasy *Data.* Atrybut *dzieci* jest zbiorem obiektów z klasy *Osoba,* a atrybut *życiorys* jest listą uporządkowanych chronologicznie obiektów z klasy *Wydarzenie.*



Rys. 1. Atrybuty obiektu klasy *Osoba*

Szczególnym obiektem kompleksowym jest **obiekt złożony** (ang. *composite object*). Obiekt złożony jest obiektem zawierającym co najmniej jedno wskazanie o semantyce kompozycji. Wskazanie takie modeluje związek fizycznego zawierania się encji w reprezentowanym fragmencie rzeczywistości. Przedstawiony powyżej obiekt kompleksowy *samochód_Waldka* jest obiektem złożonym, zawiera bowiem atrybut *silnik,* wskazujący na wystąpienie klasy *Silniki.* Wskazanie to jest wskazaniem kompozycyjnym, bowiem silnik jest częścią samochodu. Obiekt wskazywany nazywamy **obiektem-komponentem** (ang. *component object*), a wskazanie – **wskazaniem kompozycyjnym** (ang. *composite reference*). Obiekt-komponent może być również obiektem złożonym. Obiekty powiązane wskazaniami kompozycyjnymi tworzą hierarchię, nazywaną **hierarchią kompozycji** (ang. *IS-PART-OF hierarchy*). Zauważmy, że w odróżnieniu od wskazań słabych, wskazania kompozycyjne nie mogą tworzyć cykli, gdyż obiekt złożony nie może być jednocześnie swoim komponentem.

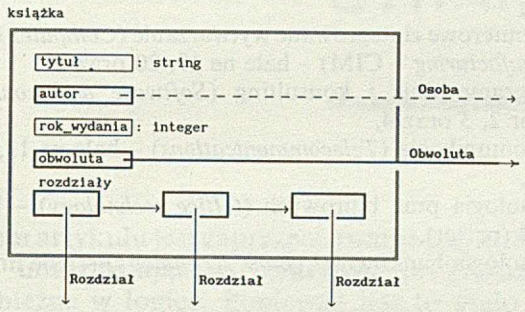
W niektórych modelach obiektowych baz danych [7] wyróżnia się wskazania kompozycyjne zależne i niezależne oraz wyłączne i współdzielone.

Wskazanie zależne (ang. *dependent reference*) jest wskazaniem na obiekt, który nie może istnieć samodzielnie. Obiekt taki może być utworzony jedynie wówczas, gdy istnieje obiekt złożony, którego jest komponentem. Natomiast usunięcie obiektu złożonego powoduje automatyczne usunięcie jego komponentów.

Wskazanie niezależne (ang. *independent reference*) jest wskazaniem na obiekt, który może istnieć samodzielnie. Jeżeli obiekt ten jest komponentem jakiegoś obiektu złożonego, to usunięcie tego obiektu złożonego nie powoduje usunięcia obiektu-komponentu.

Wskazanie wyłączne (ang. *exclusive reference*) jest wskazaniem unikalnym w bazie danych. Obiekt wskazany w sposób wyłączny jest zatem komponentem dokładnie jednego obiektu złożonego.

Wskazanie współdzielone (ang. *shared reference*) jest wskazaniem wielokrotnym. Obiekt wskazany w sposób współdzielony może być jednocześnie komponentem wielu obiektów złożonych.



Rys. 2. Typy wskazań na obiekty

Typy wskazań pokazano na rys. 2. Obiekt *książka* zawiera pięć atrybutów: *tytuł, autor, rok_wydania, obwoluta* i *rozdziały*. Atrybuty *tytuł* i *rok_wydania* są atrybutami typu prostego, atrybuty *autor* i *obwoluta* są wskazaniami na obiekty – wystąpienia odpowiednio klas *Obwoluta* i *Osoba*, a atrybut *rozdziały* jest listą wskazań na wystąpienia klasy *Rozdział*. Linią przerywaną oznaczono wskazania słabe, a linią ciągłą wskazania kompozycyjne: wyłączne (linia gruba) i współdzielone (linia cienka). Zwróćmy uwagę, że o ile rozdział może być komponentem kilku książek, to obwoluta może być komponentem dokładnie jednej książki. Jedynym wskazaniem zależnym jest wskazanie na obwolutę – nie może ona istnieć samodzielnie, tak jak autor książki lub jej poszczególne rozdziały.

Tożsamość obiektów (ang. *object identity*) polega na nadawaniu każdemu z obiektów bazy danych unikalnego identyfikatora. Unikalność identyfikatorów jest rozumiana szerzej niż zwykła niepowtarzalność w określonym zbiorze. Identyfikator obiektu przypomina obraz linii papilarnych człowieka – unikalny w danej chwili i wyróżniający go spośród innych, a jednocześnie niepowtarzalny w dowolnie odległej przyszłości. W przypadku obiektowych baz danych tożsamość obiektów oznacza, że identyfikator każdego obiektu istnieje i jest zarezerwowany dla niego od początku istnienia obiektowej bazy danych (przed utworzeniem obiektu), zostaje przypisany obiektowi z chwilą jego utworzenia, następnie reprezentuje go aż do momentu usunięcia z bazy, po czym pozostaje zablokowany, tak aby nigdy nie reprezentować żadnego innego obiektu.

Tożsamość obiektów powoduje, że istnienie obiektów jest niezależne od ich wartości. W konsekwencji mamy dwie płaszczyzny porównywania obiektów: równość i identyczność. Obiekty są równe, gdy mają te same wartości. Obiekty są identyczne, gdy mają te same identyfikatory. Równość obiektów jest cechą tymczasową, podczas gdy identyczność jest cechą stałą.

Tożsamość obiektów umożliwia współdzielenie komponentów przy konstrukcji różnych obiektów złożonych, a zatem rozszerza semantykę powiązań kompozycyjnych. Dla ilustracji wróćmy do dwóch wcześniej prezentowanych wystąpień klasy *Osoba: Waldek* i *Bożena*. Jak wynika z rys. 1, obiekty te zawierają atrybut *dzieci*, będący zbiorem wystąpień klasy *Osoba*. *Waldek* i *Bożena* mają po jednym dziecku: czteroletnim Marcinie. Gdyby obiekty nie miały tożsamości, to na podstawie atrybutu *dzieci* nie można by rozróżnić następujących dwóch sytuacji:

- *Waldek* i *Bożena* są rodzicami tego samego dziecka,
- dziecko *Waldka* jest rówieśnikiem i nosi to samo imię i nazwisko co dziecko *Bożeny*.

W zależności od sposobu konstrukcji identyfikatorów, wyróżniamy identyfikatory fizyczne i logiczne. Identyfikatory fizyczne wskazują lokalizację reprezentacji obiektu w pamięci zewnętrznej. Ich konstrukcja polega zazwyczaj na złożeniu identyfikatora wolumenu pamięci zewnętrznej, numerów cylindra, powierzchni i pierwszego sektora reprezentacji obiektu. Identyfikatory logiczne są identyfikatorami znaczącymi, będącymi łańcuchami znaków. Łańcuchy te mogą być konstruowane przez konkatenację identyfikatorów lokalnych, w sposób odzwierciedlający powiązania kompozycyjne obiektu lub przynależność do klasy.

Wersje obiektów są niezbędne, zwłaszcza w zastosowaniach obiektowych baz danych, do szeroko rozumianego komputerowego wspomagania projektowania i komputerowego wspomagania inżynierii oprogramowania [1, 11]. Z praktycznego punktu widzenia, w zastosowaniach tych jest wymagana możliwość tworzenia przez użytkowników różnych wersji samego obiektu. Część z nich ma charakter wersji historycznych, odpowiadających przejściowym etapom rozwoju projektu (programu), do których mogą być realizowane nawroty. Drugą część stanowią wersje wariantowe, przeznaczone do różnych zastosowań, dopasowane do różnych gustów i potrzeb użytkowników. Na przykład, w procesie projektowania samochodu powstaje w naturalny sposób wiele jego różnych wersji wariantowych. Z projektu bazowego są wprowadzane nowe wersje w celu np. zredukowania wymiarów samochodu, zwiększenia mocy silnika, dostosowania do przewozu towarów itp. Ponadto, są generowane różne warianty dla eksperymentowania i porównań. Wielowersyjny jest nie tylko obiekt złożony będący celem projektu (w tym przypadku samochód), lecz również większość jego komponentów, takich jak: silnik, tłoki, zawory, uszczelki, karoseria itp. Nie wszystkie wersje różnych obiektów pasują do siebie. Finalne wersje wariantowe są zatem skomponowane ze szczególnych kombinacji wersji obiektów-komponentów.

Z formalnego punktu widzenia, obiekt wielowersyjny jest częściowo uporządkowanym zbiorem własnych wersji, które reprezentują jego różne stany. Poszczególne wersje obiektu mogą być tworzone od podstaw lub wyprowadzane z innych wersji. Zależnie od przyjętego modelu, zbiór wersji obiektu może mieć strukturę drzewa, lasu lub acyklicznego grafu skierowanego.

Na zakończenie należy omówić jeszcze jedną, szczególną cechę obiektowych baz danych – możliwość dynamicznej modyfikacji hierarchii dziedziczenia klas. Języki obiektowe zazwyczaj nie umożliwiają dynamicznego wprowadzania jakichkolwiek zmian do hierarchii klas. Również klasyczne systemy baz danych mają bardzo ograniczony zbiór operacji modyfikacji schematu, np. w systemie SQL/DS są dostępne jedynie operacje tworzenia i kasowania relacji oraz dodawania nowego atrybutu do relacji. Języki obiektowych baz danych oferują znacznie bagatszy zestaw operacji modyfikowania schematu, czyli przedefiniowywania hierarchii klas. Przykładowo, w systemie ORION [2] udostępniono ponad 20 takich operacji. Można je podzielić na dwie podstawowe grupy:

- zmiany w definicji klas,
- zmiany w strukturze hierarchii klas.

Pierwsza grupa obejmuje operacje dodania do klasy nowego atrybutu lub metody, usunięcia atrybutu lub metod, zmianę

dokończenie na str. 18

CeBIT'93
HANNOVER
24. — 31. 03. 1993

Tegoroczne międzynarodowe targi technologii informatycznej i telekomunikacyjnej CeBIT odbędą się w terminie od 24 do 31 marca 1993 r. (8 dni – od środy do środy). Przy tej okazji warto przypomnieć, że CeBIT (Centrum für Büro, Information und Telekommunikation) zapoczątkowano w 1970 r. w ramach Międzynarodowych Targów Hanowerskich, jako imprezę konkurencyjną w stosunku do renomowanych wówczas francuskich i brytyjskich ekspozycji sprzętu komputerowego. Zlokalizowany w nowo zbudowanej, największej na świecie (aż do chwili obecnej!) hali targowej o powierzchni ponad 5 hektarów, CeBIT stał się dzięki dalekowzroczności organizatorów głównym czynnikiem rozwoju Targów Hanowerskich, doprowadzając je w ostatnich latach do pozycji największej na świecie imprezy targowej. W 1992 r., dysponując łącznie 478 900 m² powierzchni wystawowej brutto, targi te znacznie wyprzedziły następne w kolejności pod względem powierzchni Targi Mediolańskie (270 000 m²). Dla porównania, największe w USA Targi w Chicago dysponują powierzchnią zaledwie 174 000 m², a nasze Targi Poznańskie zajmują pod tym względem dopiero 22 miejsce (103 200 m²). Ośmielam się twierdzić, że tak szybki rozwój i niezwykły sukces rynkowy Targi Hanowerskie zawdzięczają w głównej mierze tylko CeBITowi, ponieważ ani nie miały one tradycji przedwojennych (powstały po raz pierwszy dopiero w 1947 r.), ani też nie zostały zlokalizowane w wielomilionowej aglomeracji miejskiej, takiej jak Paryż czy Londyn.

Przyspieszenie rozwoju technologii informatycznej i rewolucja mikrokomputerowa na początku lat osiemdziesiątych sprawiły, że Targi Hanowerskie zostały zdominowane przez ekspozycję CeBIT, która rozszerzyła się poza wspomnianą halę nr 1, i obejmując coraz większą liczbę hal wyparła inne gałęzie przemysłu. Spowodowało to wyodrębnienie od 1986 r. CeBIT w niezależną coroczną imprezę z terminem marcowym oraz przesunięcie podstawowej ekspozycji Targów na termin późniejszy (kwiecień).

O przygniatającej przewadze CeBIT nad innymi współczesnymi międzynarodowymi wystawami informatycznymi świadczy fakt, że zajmująca obecnie drugą na świecie pozycję impreza ORGATEC w Kolonii miała w 1990 r. tylko 142 000 m² powierzchni wystawienniczej, 1784 wystawców i 186 100 zwiedzających, a takie czołowe imprezy lat siedemdziesiątych, jak amerykański COMDEX Fall w Las Vegas w 1991 r. – 105 900 m² powierzchni, 1950 wystawców i 127 000 zwiedzających, czy słynny paryski SICOB w 1990 r. – zaledwie 50 200 m² powierzchni, 846 wystawców i 137 800 zwiedzających.

Jak poinformowali przedstawiciele Targów na konferencji prasowej 20 listopada ub.r., tegoroczna ekspozycja CeBIT zostanie rozmieszczona w 22 halach (21 w 1992 r.), połączonych w jedną całość korytarzami, co zapewnia komfortowe zwiedzenie wszystkich hal bez narażania się na kaprysy wiosennej pogody. Z własnych doświadczeń wiem, jak duże znaczenie ma takie udogodnienie dla zwiedzającego, narażonego na łatwe przeziębienie przy częstym przechodzeniu na zewnątrz z przegrzanych hal.

Podobnie jak w latach ubiegłych, całość ekspozycji podzielono na 10 ekspozycji tematycznych, które zlokalizowano w następujących halach:
- technologia informatyczna (*Information technology*) – hale nr 1, 5, 6, 7, 8 oraz 9,
- przetwarzanie sieciowe (*Network computing*) – hale nr 12 oraz 14,
- komputerowe zintegrowane wytwarzanie (*Computer integrated manufacturing* – CIM) – hale ne 19, 20 oraz 21,
- oprogramowanie i konsulting (*Software and consulting*) – hale nr 2, 3 oraz 4,
- telekomunikacja (*Telecommunications*) – hale nr 13, 16, 17 oraz 23,
- technologia prac biurowych (*Office technology*) – hale nr 1 oraz 4 (parter),
- technologia bankowości (*Bank technology*) – hale nr 2 oraz 18,
- wyposażenie zabezpieczające (*Security equipment*) – hala nr 18 (I piętro),
- badania (*Research*) – hala nr 22,
- szkolenie oraz rozwój personelu (*Training and personnel development*) – hala nr 15.

W porównaniu do 1991 r. nastąpiły znaczne zmiany w strukturze powierzchni wystawowej, odpowiadające zmianie punktów ciężkości w rozwoju technologii informatycznej. Największy wzrost powierzchni nastąpił w problematykach sieci komputerowych i badań naukowych (po + 100%), technikach C (+ 50%) oraz przetwarzania danych produkcyjnych (+ 45%), natomiast zmniejszenie – w problematyce szkolenia (— 25%), materiałów eksploatacyjnych (— 20%) oraz usług (— 10%).

Zgodnie z tradycją zostaną zorganizowane na CeBIT również ekspozycje specjalne (*Special displays*) o charakterze promocyjno-propagandowym. Będą one poświęcone następującym aktualnym tematom:
- „Centrum oprogramowania dla małych i średnich przedsiębiorstw" (*Software center for small and medium-sized companies*) – na parterze hali nr 4;
- „Oprogramowanie w Europie" (*Software in Europe*) – na I piętrze hali nr 4. Ta kolektywna ekpozycja jest organizowana po raz drugi z udziałem ok. 150 firm z krajów EWG oraz Europy Środkowo-Wschodniej i jest połączona z konkursem na najlepsze europejskie produkty programowe (nagrody „Golden Softies");
- „Biznes ze Wschodnią Europą" (*Business with Eastern Europe*) – w hali nr 6. Ekspozycja ta będzie połączona z V międzynarodowym Forum CeBIT na ten temat. Organizatorzy zaprosili kompetentnych przedstawicieli tego regionu do udziału w dyskusjach panelowych oraz prezentacji poglądów na temat aktualnego stanu i planów rozwoju informatyki i telekomunikacji;
- „Centrum użytkowników europejskich rozwiązań sieciowych w administracji komunalnej i państwowej" (*ENAC – European Network and Application Center*) – hali nr 14;
- „Systemy bankowe i finansowe" (*Banking and Finance Systems*) – na parterze hali nr 18;
- „Bezpieczny ośrodek komputerowy" (*The Safe Computer Center*) – na I piętrze hali nr 18;
- „Automatyzacja elektronicznego projektowania" (EDA – *Electronic Design Automation*) – w hali nr 21;

dokończenie na s. 25

GRZEGORZ CZAJKOWSKI
KRZYSZTOF ZIELIŃSKI
Instytut Informatyki AGH
Kraków

STRAND88 – język współbieżnego programowania w logice

Celem artykułu jest zaprezentowanie języka STRAND88 umożliwiającego efektywne programowanie współbieżne w logice. Ponieważ jest to mało znane a bardzo interesujące narzędzie programistyczne, w artykule zamieszczono wiele krótkich, specjalnie w tym celu skonstruowanych programów ilustrujących podstawowe metody programowania w języku STRAND. Ze względu na informacyjny charakter publikacji, formuła ta wydaje się być najlepszą. Oprócz pokazania możliwości zrównoleglania obliczeń i sposobów ich synchronizacji, przedstawiono także podobieństwa i różnice między językiem STRAND i Prolog.

Wybór Prologu jako języka odniesienia został podyktowany jego dużą popularnością i dostępnością oraz – co najważniejsze – niewątpliwymi związkami STRAND-u z programowaniem w logice. W tych fragmentach artykułu, które omawiają algorytm unifikacji oraz rodzaju danych, czy też w przykładach programów deterministycznych (tzn. bez możliwości znalezienia innych rozwiązań) podejście takie jest całkowicie wystarczające. Jednakże Prolog jest językiem sekwencyjnym i dlatego nie może być pomocny w zrozumieniu współbieżnych aspektów przetwarzania w STRAND-zie. Specyfiką STRAND-u jest bowiem to, że każdy nawet najprostszy program musi być pisany tak, aby mógł być poprawnie wykonany równolegle. Tak więc przy omawianiu bardzo istotnych dla STRAND-u zagadnień synchronizacji procesów zrezygnowano z odwołań do Prologu.

Do skompilowania i uruchomienia wszystkich zaprezentowanych programów wykorzystano środowisko STRAND88, wersja BUCKINGHAM RELEASE – JUNE 1990 brytyjskiej firmy *Strand Software Tchnologies Limited*. Przykłady zostały przetestowane na sieci stacji roboczych SUN4 – SPARC.

Ogólna języka charakterystyka

Typy danych. Język STRAND ma stosunkowo bardzo ubogi zbiór typów danych. Podobnie jak w Prologu, dostępne są:

- Stałe liczbowe – całkowite: 123, –123 oraz rzeczywiste: 1.23, –1.23E-2;
- Napisy: *"Napis", "napis", "jeden napis"*. Jeżeli napis rozpoczyna się od małej litery, po której występują jedynie litery, cyfry i znaki podkreślenia – można opuścić znaki cudzysłowu: *napis, jeden_napis, napis_123*;
- Zmienne: zostaną one (być może) obliczone w trakcie wykonywania programu. Zmienna rozpoczyna się od dużej litery lub od znaku podkreślenia, np.: *_zmienna, Zmienna*. Analogiczne jak w Prologu jest pojęcie zmiennej anonimowej: _.Zmiennej może być nadana wartość tylko jeden raz, co oznacza STRAND należy do grupy języków typu *single-assignment;*
- Listy: znaczenie i dostęp są identyczne jak w Prologu: lista może być albo listą pustą: [], albo mieć postać $[H|T]$, gdzie H jest jej pierwszym elementem, a T – listą. Nie istnieje żadne ograniczenie na typ obiektów listy – mogą to być dowolne elementy, np.: [123, *[a, b]*, [], *„Ostatni"*];
- Krotki: (tuples) – odpowiednik Prologowy termów. Istnieją dwie możliwości zapisu krotki:

1) w notacji prefiksowej: *student* (*Rok, Kierunek, Nazwisko*);
2) za pomocą nawiasów klamrowych: {*student, Rok, Kierunek, Nazwisko*}.

Częściej jest stosowana konwencja (2), gdyż pozwala na opuszczenie zbędnych nieraz informacji. Jeżeli wiadomo, że chodzi o krotkę opisującą studenta, zamiast pisać *student* (*Rok, Kierunek, Nazwisko*) możemy opuścić napis *student* i pisać {*Rok, Kierunek, Nazwisko*}.

Istnieją także dwa inne, nie znane w Prologu typy danych: moduły oraz *c*-strumienie. Nie będą one tu omawiane, gdyż wykorzystywanie struktury modułu nie należy do technik podstawowych, natomiast *c*-strumienie mają – zgodnie z ostrzeżeniami producenta – nie występować w przyszłych wersjach języka.

Omawiając typy danych dostępnych w STRAND-zie pojawia się pytanie, czy istnieje możliwość tworzenia cyklicznych list albo krotek (dla konkretnej implementacji Prologu podobnym pytaniem jest: czy występuje *occurs check*). W pracy [6] na stronie 3.17 stwierdzono bezwarunkowo, że nie można budować (tym bardziej – przetwarzać) struktur cyklicznych, jednakże w przypadku dostępnej wersji STRAND-u stwierdzono, że ograniczenie to nie obowiązuje. Przykład programu tworzącego i wykorzystującego taką krotkę został podany w rozdziale „Komunikacja między procesami".

Model obliczenia w STRAND-zie. Każdy proces może być przedstawiony jako krotka postaci $p(T1, .., Tn)$, gdzie p – nazwa procesu, $T1, .., Tn$ – jego argumenty, przy czym $n >= 0$. Procedurą definiującą proces p/n jest zbiór klauzul postaci:

$$p(A1, .., An) :- G1, .., Gx \mid B1, .., Bk.$$

gdzie $G1, .., Gx$ tworzą opcjonalną ($x >= 0$) sekcję prostych testów arytmetycznych, testów zgodności typów oraz sprawdzania, czy zmienna jest związana. Test nie może zmieniać wartości testowanej zmiennej. Procesy $B1, .., Bk$ ($k >= 0$) tworzą ciało procedury. Jeżeli $k = 0$ i $x > 0$, to ciało procedury jest zapisywane jako *true:*

Pracę wykonano w ramach grantu KBN nr: 8.0077.91.01

$p(A1, .., An) : - G1, .., Gx \mid true.$

Zgodnie z przyjętą w niektórych opracowaniach konwencją, *proc/num_arg* oznacza *num_arg*-argumentową procedurę o nazwie *proc*, natomiast *proc/numarg/clause_num* oznacza klauzulę o numerze *clause_num*, opisującą *num_arg*-argumentową procedurę proc.

Wykonywanie programu w STRAND-zie przebiega następująco: na początku do puli procesów zostają włożone procesy będące celami programu. Następnie niedeterministycznie jest wybierany dowolny proces i następuje próba jego zredukowania z nagłówkiem dowolnej klauzuli definiującej ten proces. Redukcji można dokonać jeżeli:

1. Jest możliwa unifikacja procesu z nagłówkiem wybranej klauzuli. Unifikacja przebiega analogicznie jak w Prologu, z jednym wyjątkiem, mogącym sprawić osobom mającym utrwalone Prologowe nawyki początkowo dużo problemów: zmiennej z puli procesów nie można nadać wartości w wyniku unifikacji. Przyjęcie tego założenia dostarcza podstawowego mechanizmu synchronizacji procesów w modelu obliczeń STRAND-u;

2. Zachodzą wszystkie relacje określone przez testy:
Warunek 1 oznacza, że program:

```
(P1)
  max(A, B, Max) :- A >= B | Max :=
  max(A, B, Max) :- B >= A | Max :=
```

Wydruk 1

poprawnie obliczy maksimum z dwóch liczb, podczas gdy ładniejszy z punktu widzenia elegancji zapisu w Prologu program:

```
(P2)
 max(A, B, A) :- A >= B | true.
 max(A, B, B) :- B >= A | true.
```

Wydruk 2

nie wykona się nigdy, jeżeli redukowanym procesem będzie np. *max*(3, 2, *Max*). Oczywiście program (P2) jest całkowicie poprawny dla np. *max*(2, 3, 3).

Jeżeli redukcja jest możliwa, na miejsce wybranego procesu do puli zostają wprowadzone procesy z ciała wybranej klauzuli, z odpowiednio zmienionymi (unifikacja!) wartościami argumentów. W wypadku, gdy nie można zredukować wybranego procesu z żadną klauzulą opisującą jego akcje, proces zostaje zwrócony bez zmian do puli, z której zostaje wybrany inny i postępowanie zostaje powtórzone. Koniec programu następuje w momencie, gdy pula procesów jest pusta. Tak więc, jeżeli Prolog nie może zredukować aktualnego celu (niepowodzenie), to wykonywanie programu zostaje przerwane (przy założeniu, że nie istnieje już możliwość nawrotu). W podobnej sytuacji STRAND zachowuje się odmiennie: jeżeli nie jest możliwa redukcja żadnego aktywnego procesu, to program wykonuje się dalej i następują kolejne, tak samo bezskuteczne próby zredukowania któregokolwiek procesu. Jeżeli więc programista chce, żeby program nie przechodził w stan totalnego zawieszenia, to powinien (o ile istnieje taka konieczność) dodawać klauzulę (klauzule) powodujące, że mimo np. nieodpowiednich argumentów program wykona się i zakończy. Program:

```
(P3)
square(N, N_squared) :- integer(N) | N_squared is N * N.
square(N, N_squared) :- otherwise  |
   display_nl({"Wrong argument type : ", N})!, N_squared := -1.
```

Wydruk 3

obliczy poprawnie kwadrat dowolnej liczby całkowitej, natomiast jeżeli pierwszym argumentem jest np. lista albo napis, to zasygnalizuje błąd i zwróci (−1) sygnalizującą innym procesom (mogą przecież istnieć procesy czekające na tę wartość), że coś nie jest w porządku. Jeżeli nie byłoby drugiej klauzuli, umieszczenie w puli procesu *square* ([], −) spowoduje, że pula nigdy nie będzie pusta, czyli wykonywanie programu nigdy się nie zakończy. Warto jest jeszcze zwrócić uwagę na dwie różnice między STRAND-em i Prologiem:

- W STRAND-zie nie jest istotna kolejność klauzul, np. jeżeli proces *max*(3, 2, *Max*) w (P1) zwróci *Max* = = 3, to nie wiadomo, czy nastąpiła redukcja z *max*/2/1, czy z *max*/2/2. Można jednak wpływać na kolejność wyboru klauzul do redukcji, np. test *otherwise* powoduje, że zawierająca go klauzula *H* jest wybrana dopiero wtedy, gdy okaże się, że nie jest i nie będzie możliwa redukcja danego procesu z żadną klauzulą opisującą ten sam co *H* proces, i poprzedzającą ją w tekście programu (przykład – P3).

- Nie istnieje nawracanie (ang. *backtracking*). Wiele osób piszących w Prologu pomyśli zapewne, że jest to całkiem sympatyczna cecha języka, ponieważ znikają chociażby problemy związane ze stosowaniem cięcia (ang. *cut*), czy zawsze zawodzącego predykatu *fail*. Powstaje jednak istotne utrudnienie, bo chcąc znaleźć wszystkie rozwiązania danego problemu, programista nie ma do dyspozycji funkcjonalnego odpowiednika Prologowego *findall*/3 (czy *setof*/3). Jeżeli więc zadaniem jest wygenerowanie pełnego zbioru rozwiązań, to program musi być napisany w odpowiedni sposób, istotnie różniący się od tego samego algorytmu zapisanego w Prologu. Odpowiednie przykłady zamieszczono w dalszej części artykułu.

STRAND jako język współbieżnego programowania w logice. Charakteryzując STRAND88 jako język współbieżnego programowania w logice należy wymienić następujące jego cechy, będące wynikiem omówionego wcześniej modelu obliczeń:

- wszystkie klauzule danej procedury są wypróbowywane niedeterministycznie (ang. *don't know nondeterminism*) aż do momentu, gdy jedna z nich okaże się odpowiednia. Jej podcele będą wykonane, a dalsze sprawdzanie i dopasowywanie innych klauzul zostaje przerwane (ang. *OR-parallelism*);
- wszystkie podcele ciała aktualnie wykonywanej klauzuli są wykonywane współbieżnie (ang. AND-parallelism);
- unifikacja dwóch obiektów jest wykonywana od lewej strony do prawej, najpierw w głąb (brak *unification-parallelism*);
- predykaty testujące (sekcja quard) nie mogą nadawać wartości zmiennym testowanym; testy nie zwiększają objętości puli procesów; programista może jednak uzupełniać zbiór predefiniowanych predykatów testujących swoimi własnymi, atomowymi procedurami;
- opisany już wcześniej brak symetrii w wykonywaniu algorytmu unifikacji.

Wymienione cechy powodują, że STRAND jest językiem co najmniej tak atrakcyjnym jak Concurrent Prolog, Parlog czy Guarded Horn Clauses [4, 7]. Nie istnieje np. konieczność deklarowania, czy zmienna będzie tylko odczytywana (ang. *read-only*) jak w Concurrent Prologu, nie trzeba też wskazywać, które zmienne są wyjściowe, a które – wejściowe, jak w Parlogu. Jednocześnie STRAND nie ustępuje wyżej wymienionym językom pod względem realizacji AND- i OR-paralelizmu.

Komunikacja między procesami

Komunikacja między procesami w STRAND-zie jest organizowana zazwyczaj przy użyciu list różnicowych lub specjalnego procesu *merger*/2 [2, 3]. Użycie tych mechanizmów zostanie pokazane na przykładzie jako dwa alternatywne sposoby zbierania wszystkich rozwiązań tego samego problemu. Wymienione techniki mają fundamentalne znaczenie dla programowania w STRAND-zie i dlatego zostanie im poświęcona osobna część artykułu.

Lista różnicowa składa się z dwóch części, A i B, gdzie $A == [E1, .., En \mid B]$ $(n >= 0)$. Do zapisu są używane różne notacje: A/B, {A, B} albo jako dwa osobne obiekty: A, B. Interpretacja tego zapisu jest następująca: do listy różnicowej $\{A, B\}$ należą elementy z A, ale nie z B. Tak więc $\{[1,2,3],[1,2,3]\}$ to lista pusta, podczas gdy do $\{[1,2 \mid X], X\}$ należą dwa elementy: 1 i 2 (o ile oczywiście X nie jest listą cykliczną: $X == [1,2 \mid X]$, albo $X == [2,1 \mid X]$ – w pierwszym przypadku $\{[1,2 \mid X], X\}$ jest pusta, w drugim – zawiera tylko 1). Listy różnicowe można, w odróżnieniu od zwyczajnych, łączyć w stałym czasie:

$$append_d_list(A/B, C/D, E/F) : -B := C, E := A, F := D,$$

a nie w czasie proporcjonalnym do długości jednej z list. Można rozważać nie tylko listy, ale bardziej ogólne struktury różnicowe [5].

Zastosowanie list różnicowych do zbierania wszystkich rozwiązań jest typowe dla języka STRAND, co zostanie pokazane na następującym przykładzie. Niech zadaniem będzie znalezienie wszystkich permutacji danej listy.

```
(P4)

go(InitList, Perm) :- perm(InitList, [], { Perm, [] }).

% perm/3/1: lista elementow do permutowania jest pusta, czyli znaleziono
%           nowa permutacje
% perm/3/2: istnieja jeszcze elementy ktore trzeba dolaczyc do permutacji
%           czesciowej SoFar
perm([], NewPerm, { A, B }) :- A := [ NewPerm | B ].
perm(List, SoFar, { A, B }) :- List =\= [] | perm([], List, SoFar, { A, B }).

% perm/4/1 - wyczerpano juz elementy rozszerzajace permutacje czesciowa -
%            lista roznicowa szukanych rozwiazan jest wobec tego pusta
% perm/4/2 - rozszerzam permutacje poczatkowa SoFar do [ H | SoFar ];
%            do tej nowej permutacji czesciowej trzeba dolozyc elementy
%            listy (Aside + T) - czyli uruchomic odpowiednie procesy perm/3
perm(_, [], _, { A, B }) :- A := B.
perm(Aside, [ H | T ], SoFar, { A, B }) :-
      append(Aside, T, NewInitList),
      perm(NewInitList, [ H | SoFar ], { A, M }),
      perm([ H | Aside ], T, SoFar, { M, B }).

append([], X, Y) :- Y := X.
append([ H | T ], X, Y) :- Y := [ H | Z ], append(T, X, Z).
```

Wydruk 4

W programie tym (wydruk 4) lista różnicowa jest inicjowana jako $\{Perm, []\}$ (*go*/2/1), gdzie *Perm*, to szukana lista wszystkich permutacji. Celem programu (P4) jest więc ukonkretnienie listy Perm. Jak jest to osiągane? Otóż, jeżeli redukowanym procesem jest *perm*/3, którego pierwszym argumentem jest [], to do redukcji zostanie wybrana klauzula *perm*/3/1 i znaleziona permutacja trafi jako jedyny element do listy różnicowej (przypisanie $A := [NewPerm \mid B]$, czyli A ma o jeden element więcej niż B). Jeżeli natomiast nastąpi redukcja z *perm*/3/2, to lista różnicowa zostanie przekazana bez zmian do *perm*/4, aby dalej zbierać rozwiązania. Klauzula *perm*/4/1 zamyka listę różnicową ($A := B$ – czyli $\{A, B\}$ jest pustą listą różnicową). Jeżeli natomiast wybraną klauzulą jest *perm*/4/2, to lista różnicowa $\{A, B\}$ (w tym momencie nie ma jeszcze żadnego powiązania między A i B, tzn. B nie jest jeszcze fizycznie ogonem A) jest przekazywana do procesów *perm*/3 i *perm*/4 jako odpowiednio $\{A, M\}$ i $\{M, B\}$. Ponieważ liśćmi drzewa generowania rozwiązania, czyli klauzulami nie produkującymi żadnych nowych procesów – oprócz atomowych operacji przypisania – są tylko *perm*/3/1 oraz *perm*/4/1 (*append*/3 nie jest interesująca, gdyż jej argumentem nie jest lista różnicowa), a więc w końcu M stanie się ogonem A oraz B będzie ogonem M. Po wykonaniu programu okaże się, że Perm jest listą zawierającą szukane permutacje o ogonie będącym listą pustą (gdyż tak zostało to zainicjowane w *go*/2/1), a więc Perm jest dokładnie tym, co miało być znalezione.

Innym podejściem do zebrania rezultatów obliczeń wykonywanych przez różne procesy jest użycie predefiniowanego (w poprzednich wersjach języka programista musiał go sam definiować) procesu *merger*/2, co pokazano w programie (P5) – wydruk 5 – na tym samym, co poprzednio przykładzie.

```
(P5)

% algorytm generowania permutacji - analogiczny jak w (P4)
go(InitList, Perm) :- merger(InStr, Perm), perm(InitList, [], InStr).

perm([], NewPerm, Str) :- Str := [ NewPerm ].
perm(List, SoFar, Str) :- perm([], List, SoFar, Str).

perm(_, [], _, Str) :- Str := [].
perm(Aside, [ H | T ], SoFar, Str) :-
       append(Aside, T, NewInitList),
       Str := [ merge(NewStr) | Str1 ],
       perm(NewInitList, [ H | SoFar ], Str1),
       perm([ H | Aside ], T, SoFar, NewStr).

% append/3 - zdefiniowane tak samo jak w (P4)
```

Wydruk 5

Użycie *merger*(InStr, OutStr) spowoduje, że wszystkie wiadomości (z wyjątkiem komunikatu *merge*/1) znajdą się w strumieniu wyjściowym *OutStr*. Jeżeli natomiast otrzymaną wiadomością jest *merge*/*NewStr*/, to zostanie otwarty nowy strumień *NewStr*. Od tej chwili do *OutStr* będą przekazywane komunikaty pochodzące z *InStr* i *NewStr*. Istotne jest to, że kolejność komunikatów z dowolnego strumienia wejściowego zostanie zachowana w *OutStr*. Przypisanie listy pustej, np. $NewStr := []$ powoduje zamknięcie kanału *NewStr*. Proces *merger*/2 zostaje zakończony wtedy, gdy są pozamykane wszystkie strumienie otwarte przez *merger*/2 i związane z Out/Str. Wtedy też następuje przypisanie OutStr := [], zamykające *OutStr*.

W programie (P5) rolę strumienia wyjściowego obsługiwanego przez *merger*/2 przyjmuje *Perm*. Jeżeli permutacja została znaleziona (*perm*/3/1), to przypisanie $Str := [Perm]$ powoduje przesłanie jej do procesu *merger*/2 (i dalej – do wyjściowej listy *Perm*) oraz zamknięcie strumienia *Str*. Analogicznie jak w (P4), w *perm*/4/1, gdzie była zamykana lista różnicowa, teraz jest zamykany strumień *Str*. Podobnie, w *perm*/4/2 przypisanie $Str := [merge(NewStr) \mid \leftarrow Str1]$ prześle odpowiedni komunikat-żądanie do *merger*/2 i w rezultacie spowoduje udostępnienie nowego, niezależnego od *Str*(*Str*1) strumienia *NewStr*. Tak więc zamiast jednego kanału *Str*, są teraz dostępne dwa. Dzięki temu nie komunikujące się ze sobą *perm*/3 i *perm*/4 będą mogły za pomocą odpowiednio *Str*1 i *NewStr*, przesyłać komunikaty do jednego strumienia.

W przykładzie programu (P4) zastosowanie list różnicowych stanowiło alternatywę wobec użycia procesu *merger*/2 (P5). Jednakże *merger*/2, odmiennie niż listy różnicowe, daje o wiele bogatsze i ciekawsze od pokazanych w programie (P5) możliwości komunikowania się i współpracy procesów. W szczególności umożliwia on sekwencjonowanie dostępu do wspólnego zasobu przez wiele procesów. Przykładem niech będzie zapowiedziany już wcześniej program tworzący strukturę cykliczną – drzewo, które może być albo drzewem pustym (*nil*) albo krotką postaci {*Numer, Przodek, LewyPotomek, PrawyPotomek*}. Istnienie elementu *Przodek* powoduje, że każde drzewo o więcej niż jednym węźle jest strukturą cykliczną, ponieważ każdy węzeł X drzewa, nie będący korzeniem, ma przodka $P(P =\= nil)$, czyli P zawiera X, a X zawiera P.

W programie (P6) (wydruk 6) drzewo jest tworzone i następnie pokazywane (*show_tr*/4). Każdy węzeł drzewa ma własny, unikalny numer-identyfikator. Możliwość generowania kolejnych, różnych liczb zapewnia proces *node_num_manager*/2, którego jedynym zadaniem jest zwracanie nowej, większej o 1 od poprzedniej, wartości. Chcąc stworzyć nowy węzeł, proces *tree_create*/4 przekazuje do swojego strumienia żądanie przydzielenia nowego numeru *get_num* (*Num*). Podobnie jak w programie (P5), komunikat *merge*(*NewInStream*) otworzy nowy dostęp do strumienia zarządzanego przez *merger*/2 i wykorzys-

tywanego przez *node_num_manager*. Dzięki użyciu procesu *merger*/2 można zatem zapewnić możliwość dwustronnej komunikacji wielu procesów z jednym.

```
(P6)

go(N, Tree) :- integer(N), N > 0 |
        start_num_manager(Input),
        tree_create(N, root, Tree, Input),
              show_tr([], Tree).
go(_, _) :- otherwise | display_nl(" Zly parametr ")!.


tree_create(0, _, Tree, InStream) :-  % 0 wezlow w poddrzewie - to jest lisc
         Tree := nil, InStream := [].  % zamknij strumien danych
tree_create(N, Ancestor, Tree, InStream) :- integer(N), N > 0 |
        N1 is N / 2, N2 is N - N1 - 1,          % ilosc wezlow w poddrzewach
            % InStream <- 1/ pobierz numer Num  2/ utworz nowy strumien
        InStream := [get_num(Num), merge(NewInStream) | Rest],
        Tree := {Num, Ancestor, Left, Right},       % utworz wezel
        tree_create(N1, Tree, Left, Rest),          % zajmij sie lewym
        tree_create(N2, Tree, Right, NewInStream).  % i prawym poddrzewem

% inicjacja procesow : merger i node_num_manager
start_num_manager(InputStream) :-
merger(InputStream, OutputStream), node_num_manager(OutputStream, 0).

node_num_manager([ get_num(Num) | Rest ], CurrentNum) :- Num := CurrentNum,
   NewCurrentNum is CurrentNum + 1, node_num_manager(Rest, NewCurrentNum).
node_num_manager([], _).
```

Wydruk 6

Proces *show_tr*/4, pozwalający na obejrzenie drzewa, pokazano w postaci programu przedstawionego na wydruku 7.

```
% pokazywanie drzewa :
% l - lewe poddrzewo, r - prawe poddrzewo, u - bezposredni przodek
show_tr([], {N, Ancestor, Left, Right}) :- display_nl(N)!,
      input({" Dokad teraz ?? ", C}, Link)!,
      decide(Link, C, {N, Ancestor, Left, Right}).

% zamiast testu tuple(X) mozna tez uzyc X =\= nil
decide([], 'u', {N, root, Left, Right}) :-
   display_nl("Jestem w korzeniu - nie moge isc wyzej !!! ", Link)!,
   show_tr(Link, {N, root, Left, Right}).
decide([], 'u', {_, Ancestor, _, _}) :- tuple(Ancestor) | show_tr([], Ancestor
decide([], 'r', {N, Ancestor, Left, nil}) :-
   display_nl(" Prawe poddrzewo to lisc - nie moge tam zejsc !!! ", Link)!,
   show_tr(Link, {N, Ancestor, Left, nil}).
decide([], 'r', {_, _, _, R}) :- tuple(R) | show_tr([], R).
decide([], 'l', {N, Ancestor, nil, Right}) :-
   display_nl(" Lewe poddrzewo to lisc - nie moge tam zejsc !!! ", Link)!,
   show_tr(Link, {N, Ancestor,nil, Right}).
decide([], \l{}', {_, _, L, _}) :- tuple(L) | show_tr([], L).
decide([], _, _) :- otherwise | true.
```

Wydruk 7

Synchronizacja procesów

Z modelu wykonywania programu w STRAND-zie wynika, że jest on właśnie wykonywany współbieżnie. Stąd zasadnicze znaczenie ma właściwa synchronizacja procesów. Ukazuje się ją przez wykorzystanie procesu *merger*/2, tak jak to pokazano wcześniej albo stosując jedną z konstrukcji, opartą na zasadzie zawieszenia redukcji procesu do momentu ukonkretnienia jego zmiennej w przypadku konieczności jej unifikacji ze stałą w klauzuli, co jest podstawowym założeniem modelu obliczeniowego STRAND-u.

Bardzo prostą konstrukcją, wykorzystującą tę zasadę, jest tzw. metoda *short circuit* [2]. Polega ona na takim połączeniu współpracujących procesów, aby dowolne dwa albo miały wspólną zmienną, albo aby były połączone łańcuchem zmiennych pośrednio, a więc przez inne procesy. Przykładem (wydruk 8) niech będzie bardzo prosty program różniczkujący symbolicznie nieskomplikowane wyrażenia (przy okazji można zwrócić uwagę na łatwość przetwarzania symbolicznego w STRAND-zie).

Sam algorytm różniczkowania nie wymaga komentarzy. Obliczanie pochodnej może trochę potrwać (np. $x^* ... (400\ x\text{-ów}) .. *x$), albo nawet – w przypadku dużej złośliwości osoby testującej – nigdy się nie zakończyć (np. dla $X == x^* X$). Niezależnie jednak od czasu wykonywania się programu,

```
(P6)

go(InExpr) :- diff(InExpr, OutExpr),
              ground(OutExpr, ground, Ground),
              write_result(Ground, OutExpr).

diff(x, OutExpr) :- OutExpr := 1.
diff(Int, OutExpr) :- integer(Int) | OutExpr := 0.
diff(sin(X), OutExpr) :-  diff(X, OutX), OutExpr := OutX * cos(X).
diff(cos(X), OutExpr) :-  diff(X, OutX), OutExpr := OutX * (- sin(X) ).
diff(A + B, OutExpr) :- diff(A, AOut), diff(B, BOut), OutExpr := AOut + BOut.
diff(A * B, OutExpr) :- diff(A, AOut), diff(B, BOut),
                        OutExpr := A * BOut + AOut * B.
diff(X, Y) :- unknown(X) | Y := bad_arg.
diff(_, Y) :- otherwise | Y := bad_arg.

% ground/3:
%    1> sprawdzanie operatorow dwuargumentowych ( '+' i '*')
%    2> sprawdzanie operatorow unarnych ( 'sin', 'cos', '-' )
%    3 i 4> natychmiastowa propagacja wartosci LIn do LOut
%    5> przerwania lancucha procesow przekazujacych aktualna wartosc
%       LIn - od tej pory przekazywane bedzie bad_arg
ground({ Op, A1, A2 }, LIn, LOut) :- data(Op) |
        ground(A1, LIn, LM), ground(A2, LM, LOut).
ground({ Op, A }, LIn, LOut) :- data(Op) | ground(A, LIn, LOut).
ground(Int, LIn, LOut) :- integer(Int) | LOut := LIn.
ground(x, LIn, LOut) :- LOut := LIn.
ground(bad_arg, _, LOut) :- LOut := bad_arg.
write_result(ground, Expr) :- display_nl(Expr)!.
write_result(bad_arg, _) :- display_nl(' Zly argument !!!')!.
```

Wydruk 8

zawsze jest możliwy wybór do redukcji procesu *write_result*/2, zanim wyrażenie *InExpr* zostanie opracowane, co – w przypadku braku synchronizacji spowodowałoby wypisanie adresów zmiennych zamiast konkretnych wartości. Program (P6) prezentuje zastosowanie metody *short circuit* jako sposobu wstrzymania wykonania procesu *write_result*/2 aż do chwili, gdy zostanie obliczona pochodna.

Współbieżnie z obliczaniem pochodnej jest sprawdzane ukonkretnienie wyniku. Dokonujący tego proces, *ground*/3, przekazuje wartość drugiego argumentu trzeciemu, jeżeli pierwszy jest ukonkretniony. Ponieważ *ground*/3/1 uaktywnia dwa procesy *ground*/3, jest konieczne wprowadzenie zmiennej pośredniej *LM*. Proces *ground*/3 czeka na ukonkretnienie pierwszego argumentu, dlatego też procesy różniczkujące muszą nadać wynikowi jakąś wartość (tutaj: *bad_arg*), nawet jeżeli wyrażenie do zróżniczkowania jest niepoprawne. Po napotkaniu *bad_arg* jako swojego argumentu, *ground*/3 przekazuje dalej nie *LIn* (która z kolei też mogła już mieć nadaną wartość *bad_arg* – przecież błąd w wyrażeniu można zrobić w kilku miejscach), ale stałą *bad_arg*, odpowiednio interpretowaną przez *write_result*/2.

Programowanie równoległe

STRAND może być uruchomiony na wielu komputerach połączonych siecią. Wieloprocesorowe wersje programów, pozwalające na fizyczne zrównoleglenie obliczeń, różnią się nieznacznie od swoich jednoprocesorowych wersji – programista wybiera topologię oraz określa, które procesy mają być wykonywane na określonych procesorach.

Dokładny opis możliwości i sposobów programowania równoległego przekraczałby objętość tego artykułu (szczegóły w [6]), dlatego też zostanie rozpatrzony jedynie przypadek, gdy programista za pomocą dyrektywy *machine* (*Topology*) z góry definiuje topologię systemu rozproszonego (nie zmienia się ona dla żadnego procesu w trakcie wykonywania programu). Do dyspozycji są następujące rodzaje topologii:

- *ring* – każdy procesor jest połączony z dwoma innymi, oznaczanymi *bwd* i *fwd;*
- *connected* – każdy procesor jest połączony ze wszystkimi innymi; odwoływanie się do poszczególnych węzłów sieci następuje przez podanie numeru N danego procesora (odpowiedni proces zostanie przesłany do procesora o numerze $N1 := = N \bmod$ – liczba procesorów);
- *torus* – każdy procesor jest połączony z czterema innymi,

oznaczanymi *east, north, west, south;* ten rodzaj topologii może być traktowany jako rodzaj nieograniczonej siatki. Idąc przez cały czas w jednym wybranym kierunku z czterech dopuszczalnych, po N krokach ($N ==$ liczba procesorów) zostaną odwiedzone wszystkie procesory, a wędrujący proces znajdzie się tam, skąd wyruszył.

Chcąc uruchomić proces *remote_process* na innym procesorze, np. w topologii *torus*, wystarczy napisać: *remote_process @ east*.

Jeżeli wymagana liczba procesorów nie jest dostępna, to np. w systemie trzech połączonych komputerów o nazwach: *tulip, rose* i *crocus* po wywołaniu STRAND-u z opcją: *st -c tulip tulip rose rose crocus* będzie możliwe symulowanie obliczeń na sześciu procesorach: *crocus* będzie jednym procesorem, *rose* będzie traktowany jako dwa wirtualne procesory, akomputer *tulip* – jako trzy (przy założeniu, że wywołanie nastąpiło z tulipa). Biblioteka topologii wirtualnych zapewnia więc możliwość symulowania dostępu do dowolnie dużej liczby procesorów.

Poniższy przykład prezentuje program przeszukujący graf spójny metodą *najpierw wszerz* (ang. *breadth first*), obliczany na topologii typu *torus* (pomysł na program P7) można znaleźć w [1].

```
-machine(torus).

% na poczatku drzewo przeszukiwania to tylko lisc
go(StartNode, Paths) :- bfs(leaf(StartNode), { Paths, [] }).

% bfs/2/1: drzewo w grafie pozostale do przeszukania jest puste -> koniec
% bfs/2/2: drzewo jeszcze niepuste - trzeba je sprawdzic
bfs(nil_tree, { A, B}) :- A := B.
bfs(Tree, { A, B }) :- Tree =\= nil_tree |
   expand([], Tree, NewTree, { A, M }), bfs(NewTree, { M, B }).

% expand/4 - zwieksza glebokosc budowanego drzewa o 1
% expand/4/1: znaleziono rozwiazanie - zwrocona wartosc nil_tree spowoduje
%   odciecie koncowej czesci Path - tej bez zadnych mozliwych rozgalezien;
%   rozwiazanie jest zapamietywane w liscie roznicowej
% expand/4/2: lisc drzewa ne zawiera wierzcholka c - lista roznicowa staje
%   sie pusta, a N staje sie wezlem wewnetrznym drzewa poszukiwan rozwiazan
% expand/4/3: przegladany jest wierzcholek wewnetrzny drzewa -
%   expandall/5 przejrzy i zmodyfikuje wszystkie poddrzewa
expand(Path, leaf(c), NewTree, { A, B }) :-
   NewTree := nil_tree,
   A := [ [ c | Path ] | B ].
expand(Path, leaf(N), NewTree, { A, B }) :- N =\= c |
   A := B,
   nodes:nodes(N, Nodes), % skompilowany modul nodes zawiera opis grafu -
   new_tree(N, NewNodes, NewTree).
expand(Path, node(N, List), NewTree, { A, B }) :-
   expandall([ N | Path ], List, [], NewTree, { A, B }).

% expandall/5/1 - stosuje procedure expand/4 do listy drzew (drugi argument)
% expandall/5/2: przynajmniej jedno poddrzewo ze sprawdzonej listy
%   nie zostalo odrzucone - zwracane jest nowe drzewo
% expandall/5/3: wszystkie poddrzewa z zostaly odrzucone - zwracana jest
%   wartosc nil_tree
expandall(Path, [ H | T ], Accepted, NewTree, { A, B }) :-
   expand(Path, H, NewH, { A, M })@east,
   decide(NewH, Accepted, Accepted1),
   expandall(Path, T, Accepted1, NewTree, { M, B })@north.
expandall([ H | _ ], [], Accepted, NewTree, { A, B }) :- Accepted =\= [] |
   NewTree := node(H, Accepted), A := B.
expandall(_, [], [], NewTree, { A, B }) :- A := B, NewTree := nil_tree.

decide(nil_tree, Accepted, Accepted1) :- Accepted1 := Accepted.
decide(Tree, Accepted, Accepted1) :- Tree =\= nil_tree |
   Accepted1 := [ Tree | Accepted ].

new_tree(_, [], NewTree) :- NewTree := nil_tree.
new_tree(Node, List, NewTree) :- List =\= [] | NewTree := node(Node, List).

% wybor tych elementow pierwszej listy, ktore nie wystepuja w drugiej
new_leaves([], _, Selected) :- Selected := [].
new_leaves([ H | T ], List, Selected) :-
   new_leaves([], _, Selected) :- Selected := [].
   member(H, List, Result),
   accept(Result, H, Selected, Selected1),
   new_leaves(T, List, Selected1).

accept(false, H, Selected, Selected1) :- Selected := [ leaf(H) | Selected1 ].
accept(true, _, Selected, Selected1) :- Selected1 := Selected.

member(_, [], Result) :- Result := false.
member(H, [ H | _ ], Result) :- Result := true.
member(X, [ _ | T ], Result) :- otherwise | member(X, T, Result).
```

Wydruk 9

W programie tym (wydruk 9), oprócz zbierania ścieżek w grafie od podanego wierzchołka startowego do *c*, zadaniem jest także takie zsynchronizowanie procesów, aby poszukiwanie rozwiązania o długości $(N + 1)$ nie rozpoczęło się, zanim nie zostaną znalezione wszystkie rozwiązania długości (N).

Program ten można nazwać programem ukierunkowanym na przetwarzanie struktur danych (ang. *data-oriented*). Budowane w grafie drzewo poszukiwań jest powiększane o jeden poziom (*expand*/4), przy czym są odcinane gałęzie nie mogące zapewnić sukcesu przy następnym przeglądaniu drzewa. Wykonywanie programu kończy się wtedy, gdy wyczerpano wszystkie możliwości wyboru ścieżek (*bfs*/2/1).

Do zapewnienia wymaganej synchronizacji w programie (P7) wykorzystano metodę wynikającą bezpośrednio ze STRAND-owego modelu wykonywania programu i przebiegu unifikacji w tym języku. Przebywający w puli procesów *bfs*/2 nie będzie zredukowany z żadną klauzulą, dopóki nie zostanie ustalona wartość jego pierwszego argumentu. Ale nowe drzewo nie zostanie utworzone, zanim poprzednie nie zostanie gruntownie przebadane. Wynika to z faktu, że *expandall*/5 tworzy nowe (być może puste) drzewo dopiero po otrzymaniu wyników przeszukiwania wszystkich poddrzew. Tak więc dostępność danych decyduje o kolejności redukowania i uaktywniania procesów.

Z programu (P7) łatwo jest utworzyć program wykonujący się na jednym procesorze przez usunięcie adnotacji @. Odwrotnie, można też w prosty sposób zrównoleglić program (P4) czy (P5). Poprawnie napisane programy w STRAND-zie można uważać za programy równoległe – bez względu na to, czy w ich tekście znajdują się odwołania do innych procesorów, czy też nie.

* * *

Ze względu na ograniczoność miejsca nie przedstawiono innych istotnych zagadnień związanych ze STRAND-em, które zostaną krótko omówione poniżej.

Istnieje – choć nie na taką skalę w Prologu – pewna odpowiedniość między strukturami danych (tutaj: krotkami) a procedurami. Można wyznaczać nazwę procedury do wywołania dynamicznie, w trakcie wykonywania programu. Jest to bardzo duże ułatwienie przy pisaniu np. programów zapewniających rozproszenie obliczeń w sieci procesorów. Lista zadań do wykonania składa się z krotek będących nagłówkami procesów, które są rozsyłane na poszczególne procesory i tam wykonywane bez jakichkolwiek modyfikacji. Nie jest konieczne pisanie osobnej klauzuli dla każdego możliwego rodzaju przesłanego zadania, bo nadchodząca krotka jest wykonywana za pomocą systemowego procesu *spawn*/1 lub *spawn*/2.

Bardzo użyteczną cechą języka jest łatwość dołączania procedur napisanych w językach C i Fortranie (STRAND88 jako tzw. *glue language*). Pozwala to użyć już istniejący kod a także zrównoleglić programy sekwencyjne. Odpowiednie zastępowanie procedur STRAND-owych ich odpowiednikami w językach imperatywnych może znacznie przyspieszać wykonywanie programów, gdyż tak, jak każdy prologopodobny język, również i STRAND nie nadaje się bezpośrednio do niektórych zastosowań. Na przykład, program odejmujący od zadanej liczby naturalnej wartość 1 aż do osiągnięcia zera wykonuje się średnio 25 razy dłużej niż program, w którym obejmującą pętlę realizuje dołączona funkcja napisana w języku C. W doświadczeniu polegającym na obliczaniu kolorów dla fraktala Mandelbrota STRAND-owy program zarządzający procedurami w C, uruchomiony na homogenicznej sieci dziewięciu procesorów (SUN SLC) wykonywał się 35,04 sekund, podczas gdy wykonanie na jednym procesorze zajęło 174 sekundy. Takie, a nie inne (np.

dziewięciokrotne) przyspieszenie wynika głównie z faktu przesyłania znacznych ilości informacji między procesorami.

Osobnym zagadnieniem jest wydajność przetwarzania rozproszonego. Jeżeli wykorzystanie wieloprocesorowych możliwości STRAND-u ma mieć w danej aplikacji znaczenie nie tylko teoretyczne czy poznawcze (jak w tym artykule), lecz także praktyczne, przynoszące wymierne zyski czasowe w stosunku do obliczeń jednoprocesorowych, warto użyć dostarczonego przez producenta profilera.

Można stwierdzić, że STRAND88 jest interesującym, godnym polecenia produktem, wymagającym od programisty innego podejścia niż klasyczne języki programowania, jak C czy Pascal, ale także dającym możliwości inne niż te języki.

LITERATURA

[1] Bratko I.: Prolog Programming for Artificial Inteligence. Adison Wesley, 1986
[2] Foster I., Taylor S.: STRAND – New Concepts in Parallel Programming. Prentice-Hall, Inc., 1990
[3] Foster I., Taylor S.: A practical Parallel Programming Tool. North American Conference on Logic Programming, MIT Press 1989
[4] Keith L. Clark: Parallel Logic Programming. The Computer Journal, Vol. 33, No. 6, 1990
[5] Sterling L., Shapiro E.: The art of Prolog. MIT Press Cambridge, 1986
[6] Strand Software Technologies Limited: STRAND'88 USER MANUAL BUCKINGHAM RELEASE, June 1990
[7] Taylor S.: Parallel Logic Programming Techniques. Prentice-Hall, 1989,

meditronik SPÓŁKA z o.o.

UMC UNITED MICROELECTRONICS CORPORATION

HEWLETT PACKARD COMPONENTS

BOURNS

- CZĘŚCI ELEKTRONICZNE
- KOMPUTERY PS/1,PS/2
- DRUKARKI HP
- INSTALACJE SIECI KOMPUTEROWYCH

Partnerzy handlowi: ANALOG DEVICES, ITT, MOTOROLA, SAMSUNG, TELEFUNKEN i inni

- UKŁADY PAMIĘCI
- UKŁADY KOMPUTEROWE
- UKŁADY KOMUNIKACYJNE I KOMERCYJNE

PRZEDSTAWICIELSTWO

- TRANSOPTORY
- WSKAŹNIKI ŚWIETLNE
- WYŚWIETLACZE LED
- PRODUKTY KODÓW KRESKOWYCH
- KONTROLERY I CZUJNIKI RUCHU
- TECHNIKA ŚWIATŁOWODOWA
- ELEMENTY W.CZ. I MIKROFALOWE
- PODZESPOŁY DO MONTAŻU POWIERZCHNIOWEGO (SMD)

DYSTRYBUCJA

- POTENCJOMETRY TRIMPOT
- HYBRYDY REZYSTOROWE
- REZYSTORY SUBMINIATUROWE
- BEZPIECZNIKI MULTIFUSE
- POTENCJOMETRY PRECYZYJNE
- POTENCJOMETRY PANELI CZOŁOWYCH I KODERY
- CEWKI I TRANSFORMATORY
- CZUJNIKI CIŚNIENIA, POŁOŻENIA I PRZYŚPIESZENIA

PMI

DYSTRYBUCJA

IBM Business Partner

meditronik sp. z o.o.

00-194 Warszawa,ul. Dzika 4
tel. (02) 6352263, 6352264
fax (02) 6352195, tlx 816075

SO/575/91

Obiektowe bazy danych

dokończenie ze str. 11

definicji atrybutu, zmianę kodu metody, zmianę źródła dziedziczenia atrybutu lub metody itp. Druga grupa obejmuje operacje: dodania nowej klasy, usunięcia klasy, dodania lub usunięcia klasy z listy nadklas danej klasy, zmianę uporządkowania listy nadklas danej klasy itp.

Składnia i semantyka operacji modyfikacji hierarchii klas jest zgodna z wymaganiami podejścia obiektowego i polega na przesłaniu komunikatu z odpowiednimi argumentami do obiektu klasowego, którego dotyczą modyfikacje.

LITERATURA

[1] Adiba M.: Histories and Versions for Multimedia Complex Objects. IEEE Data Engineering Bulletin, Dec, 1988, pp. 1–8
[2] Banerjee J., Kim W., Kim H., Korth H.: Semantics and Implementation of Schema Evolution in Object-Oriented Databases. Proc. ACM SIGMOD Conf., San Francisco, May 1987
[3] Caruso G., Sciore F.: The VISION Object-Oriented Database Management System. Proc. of Works, on Database Programming Languages, Roscoff, France, 1987
[4] Carey M., DeWitt D., Vandenberg S.: A Data Model and Query Language for EXODUS. ACM SIGMOD Proc., Chicago, Illinois, 1988
[5] Fishman D. et. al.: IRIS: an object-oriented database management system. ACM TOIS, 1986
[6] Hudson S. E., King R.: CACTIS: A Database System for Specifing Functionally Defined Data. Proc. of Works. on Object-Oriented Databases, Pacific Grove, California, 1986
[7] Kim W., et al.: Composite Object Support in an Object-Oriented Database System. Proc. of Conf. on Object-Oriented Programming Systems, Languages and Applications, Orlando, Florida, 1987
[8] Lecluse C., Richard P., Velez F.: O_2, an Object-Oriented Data Model. ACM-SIGMOD Proc., Chicago, Illinois, USA, 1988
[9] Maier D.: The Theory of Relational Databases. Computer Science Press, 1983
[10] Vbase Integrated Object System, Technical Overview. Ontologic Inc., 47 Mannings Road, Billarica MA, 1987
[11] Zdonik S. B.: Version Management in an Object-Oriented Database. Int. Works. on Advanced Programming Environments, Norway, 1986, pp. 139–200.

Od sieci neuronowych do neurokomputerów

dokończenie ze str. 6

LITERATURA

[1] Arsenalut H. H., Macuków B.: Beyond Pattern Recognition with Neural Nets, SPIE, 960, 1988, pp. 206–216
[2] Berus T., Macukow B.: Od sieci neuronowych do neurokomputerów, Informatyka nr 4, 1991, s. 14–17
[3] Carpenter G. A., Grossberg S.: The ART of Adaptive Pattern Recognition, IEEE Computer, 21. March 1988
[4] Kohonen T.: Self Organization and Associative Memory, Springer-Verlag, 1990
[5] Macukow B., Arsenault H. H.: Logic Operations with Neural Networks, Control and Cybernetics, 20, 1991, pp. 115–133
[6] von der Malsburg C.: Self-Organization of Orientation Sensitive Cells ib the Striate Cortex, Kybernetik, 14, 1973, pp. 85–100
[7] Minsky M., Pappert S.: Perceptrons: An Introduction to Computational Geometry, MIT Press, 1969.

Prenumerata to pewność otrzymywania INFORMATYKI!

DARIUSZ CABAN
Instytut Informatyki
Teoretycznej i Stosowanej PAN
Gliwice

Środki łączności w przemysłowych rozproszonych systemach sterowania

Sterowanie procesami technologicznymi wymaga przetwarzania dużej ilości informacji. Jak podano w pracy [12] np. w typowej elektrowni konwencjonalnej liczba dwustanowych i analogowych zmiennych procesowych wynosi ponad 10 000, liczba punktów pomiarowych – ok. 8000, liczba urządzeń wykonawczych – ok. 1500. Zcentralizowany system zbierający i przetwarzający dane ze wszystkich punktów pomiarowych nie jest w stanie realizować zadań sterowania w sposób efektywny. Napływ do komputera centralnego tak dużej ilości informacji, którą należy przetworzyć w krótkim odcinku czasu, a jednocześnie konieczność szybkiej reakcji na sytuacje wyjątkowe (np. przekroczenie dopuszczalnej wartości zmiennej procesowej) może spowodować jego przeciążenie, co w ekstremalnym przypadku prowadzi do utraty kontroli nad sterowanym obiektem. Uszkodzenie komputera pociąga za sobą unieruchomienie całego systemu. Stąd w nowoczesnych rozwiązaniach systemów sterowania dominują struktury rozproszone, wielokomputerowe. Za tego typu rozwiązaniami przemawia również to, że:

- automatyzowane obiekty są zazwyczaj obiektami rozległymi w sensie przestrzennym; okablowanie takiego obiektu dla sprzężenia z komputerem centralnym byłoby bardzo kosztowne,
- ich struktura jest z natury hierachiczna [16].

Przesyłanie dużej ilości informacji o procesie, w pewnych odstępach czasu (zależnie od specyfiki procesu może on wynosić zarówno 1 s jak i 1 h), wymaga użycia odpowiednich środków łączności. Nie tylko mediów transmisyjnych o odpowiedniej przepustowości i dużej odporności na zakłócenia, powstające w środowisku przemysłowym, ale także takiej organizacji transmisji, aby zapewnić poprawność przesyłanych danych. Pojęcie środków łączności obejmuje zatem kilka elementów:

- media transmisyjne,
- protokoły komunikacyjne,
- metody dostępu do medium transmisyjnego.

Mgr inż. DARIUSZ CABAN ukończył w 1991 r. studia o specjalności Budowa i Oprogramowanie Maszyn Matematycznych na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej w Gliwicach. W IITiS PAN współuczestniczy w pracach nad syntezą systemów rozproszonych. Jego zainteresowania zawodowe obejmują zagadnienia związane z mikrokomputerowymi systemami czasu rzeczywistego (sprzęt i oprogramowanie) oraz transmisją danych.

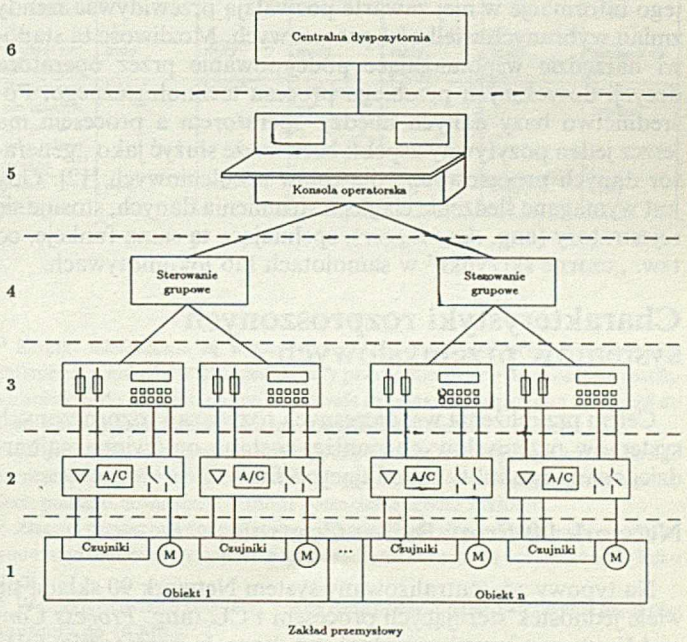
W artykule omówiono środki łączności stosowane w przemysłowych rozproszonych systemach sterowania. Wykorzystano materiały firmowe kilku światowych producentów takich systemów, jak: Bailey Controls (Network 90, INFI-90), Gould Inc. (Modbus), AEG (GeΔmatics) oraz informacje z publikacji książkowych i artykułów specjalistycznych.

Struktura rozproszonych systemów sterowania

Hierarchiczna struktura systemów sterowania to jedna z przyczyn stosowania w tych systemach układów wielokomputerowych [16]. Wyróżnia się sześć poziomów [12]:

- sterowany obiekt,
- interfejs z obiektem i sterowanie najniższego poziomu,
- sterowanie lokalne i zabezpieczenie tej części obiektu,
- sterowanie grupowe,
- udostępnianie informacji o stanie procesu, podejmowanie decyzji co do dalszego jego przebiegu,
- koordynacja pracy poszczególnych obiektów.

Strukturę przykładowego systemu sterowania przedstawiono na rysunku 1.



Rys. 1. Hierarchiczna struktura rozproszonego systemu sterowania

Najniższy poziom stanowi sterowany obiekt wraz z zainstalowanym na nim zespołem czujników i urządzeniami wyko-

nawczymi. Na kolejnym poziomie znajdują się elementy sterujące urządzeniami wykonawczymi oraz przetworniki wielkości nieelektrycznych na sygnały elektryczne. Tu sygnały analogowe podlegają również konwersji na postać cyfrową. Informacja ta zostaje, na następnym poziomie, wykorzystana do celów automatycznego sterowania i regulacji oraz zabezpieczenia pewnej wydzielonej części obiektu (ang. *local closed-loop and protection operations*). Obróbka napływających z obiektu danych musi być przeprowadzana w czasie rzeczywistym i jest w większości przypadków realizowana przez specjalizowane procesory cyfrowe. Koordynacja pracy od kilku- do kilkunastu lokalnych jednostek sterujących odbywa się na kolejnym poziomie systemu. Dawniej dokonywana w trybie ręcznym, dzisiaj przeważnie pół- lub w pełni automatycznie. Z wyższego poziomu, sterującego pracą kilku grup, informacja o stanie procesu jest dostarczana operatorowi oraz mogą być wysyłane polecenia do urządzeń wykonawczych. Przetwarzaniem danych zajmują się tutaj komputery o dużej mocy obliczeniowej. Na najwyższym poziomie struktury koordynuje się działanie kilku obiektów, dokonuje rozdziału zadań i obciążeń między obiektami, planuje produkcję w ramach zakładu przemysłowego.

Dane o procesie są udostępniane operatorowi na piątym poziomie systemu. Ma on oczywiście możliwość ingerencji również na niższych poziomach, np. sterowanie automatyczne można zastąpić sterowaniem ręcznym (producenci systemów dostarczają stacyjki sterowania ręcznego), ale tylko w wyjątkowych przypadkach. Operator powinien być informowany o stanie procesu na bieżąco (w czasie rzeczywistym). W rzeczywistości dane te są dostarczane z pewną zwłoką (ang. *data absolescence*), wynikającą z opóźnień komunikacyjnych (ang. *communication delays*) oraz czasu koniecznego na wizualizację stanu procesu na ekranie monitora (ang. *screen buid-up time*). W praktyce nie stosuje się rozwiązań pozwalających operatorowi na bezpośredni dostęp do urządzeń na drugim poziomie (sensomotorial), bowiem wskutek opóźnień komunikacyjnych pobranie żądanych danych trwałoby zbyt długo [12]. Stan procesu jest przechowywany w bazie danych, uaktualnianej i uzupełnianej bądź cyklicznie, bądź po wystąpieniu określonych zdarzeń, bądź na żądanie. Dzięki temu rozwiązaniu uzyskuje się zmniejszenie czasu potrzebnego na wizualizację, prowadzi to jednak do znacznego obciążenia linii komunikacyjnych. Oprócz tego informacje w niej zawarte pozwalają przewidywać trendy zmian wybranych wielkości procesowych. Możliwość ta stanowi narzędzie wspomagające podejmowanie przez operatora decyzji dotyczących przebiegu procesu technologicznego. Pośrednictwo bazy danych między operatorem a procesem ma jeszcz jeden pozytywny aspekt: baza może służyć jako „generator danych procesowych" dla celów szkoleniowych [12]. Gdy jest wymagane śledzenie ciągłego strumienia danych, stosuje się rejestratory (ang. *data logger*), spełniające tą samą funkcję, co tzw. „czarne skrzynki" w samolotach lub lokomotywach.

Charakterystyki rozproszonych systemów przemysłowych

Celem przybliżenia współczesnych rozwiązań rozproszonych systemów przemysłowych poniżej zostaną omówione najbardziej znane produkty przodujących firm.

Network 90 firmy Bailey Controls

Na typowy zdecentralizowany system Network 90 składa się wiele jednostek sterujących procesem PCU (ang. *Process Control Unit*) oraz innych urządzeń, np. konsola operatorska OIU (ang. *Operator Interface Unit*), kanał przemysłowy z komputerem CIU (ang. *Computer Interface Unit*), połączonych w sieć o konfiguracji pierścieniowej. W charakterze medium transmisyjnego są wykorzystane kable koncentryczne.

PCU jest przeznaczona do pobierania i przetwarzania informacji o procesie w pobliżu miejsca ich powstawania. Zawiera moduły:

- wejść-wyjść dla sprzęgnięcia ze sterowanym procesem,
- przetwarzające dane procesowe, oparte na mikroprocesorach 8- i 16-bitowych,
- sprzęgające z siecią komunikacyjną,

oraz własny system zasilania. Moduły komunikują się przez wspólną magistralę modułową (ang. *module bus*). Dane procesowe są udostępniane operatorowi systemu na konsolach operatorskich, wyposażonych w monitory graficzne, klawiatury oraz środki pozwalające przechowywać te dane na trwałych nośnikach (drukarki, dyski elastyczne). Jednostki sprzęgające z komputerem CIU umożliwiają włączenie do systemu komputerów różnych typów i pochodzących od różnych producentów. Tym samym mogą zostać wykorzystane zarówno duże komputery typu *mainframe* jak i przenośne komputery osobiste. Komunikacja między CIU a komputerem odbywa się przez interfejs szeregowy RS422/RS232C w trybie dwukierunkowym. Podwyższa to elastyczność systemu, gdyż środki te pozwalają na użycie do realizacji zadań związanych ze sterowaniem procesem pakietów programowych, zaimplementowanych na komputerach komercyjnych [4].

INFI 90 firmy Bailey Controls

INFI 90 jest wersją rozwojową systemu Network 90. Oprócz kabli koncentrycznych, w charakterze medium transmisyjnego, można wykorzystać światłowody. W systemie tym firma Bailey wprowadziła nową rodzinę modułów przetwarzających (składników PCU) opartą na mikroprocesorach 32-bitowych. INFI 90 może zostać sprzęgnięty zarówno z już istniejącymi systemami Network 90 jak i, przez interfejs RS232C, z komputerami różnych typów. Ponadto producent oferuje sprzęgi z siecią ETHERNET [3].

Modbus firmy Gould Inc.

Jest to system o organizacji typu *Master-Slave*. Oznacza to, że w systemie istnieje jedna wyszczególniona jednostka (komputer, terminal inteligentny, niektóre typy sterowników programowalnych firmy Gould) zarządzająca pracą całego systemu i wymianą informacji między jego komponentami. Jako medium transmisyjne można wykorzystać: skrętkę, kable koncentryczne, światłowody, środki telefonii publicznej, środki komunikacji radiowej i satelitarnej. Firma Gould dostarcza wiele typów sterowników programowalnych PLC oraz modułów wejść-wyjść analogowych i cyfrowych. Ponadto firma ta oferuje oprogramowanie komunikacyjne w wersji źródłowej (napisane w języku Fortran IV), które ułatwia implementację protokołu komunikacyjnego. Tworzenie oprogramowania odbywa się na komputerze Master, skąd program wynikowy jest przesyłany do pamięci sterownika [10].

W Europie system ten, na licencji firmy Gould Inc., produkuje firma AEG.

GeΔmatics firmy AEG

Jest to hierarchiczny system o konfiguracji magistralowej. Na poziomie najniższym (MODNET 1) zrealizowano sprzęgnięcie ze sterowanym procesem. Komunikację między układami wejścia-wyjścia i ich modułami nadrzędnymi oparto na standardzie RS485 oraz protokole SDLC. Komunikacja z procesem odbywa się na poziomie 2 (MODNET 2) przy użyciu sterowników programowalnych oraz komputerów procesowych czasu rzeczywistego. Wyższy poziom stanowią komputery nadzorujące pracę poszczególnych obiektów oraz komputery z centralnego

ośrodka dyspozycyjnego, realizujące zadania planowania, zarządzania itp. Poziom ten (MODNET 3) stanowi tzw. sieć podstawową (ang. *backbone network*). Do sieci mogą być przyłączone zarówno urządzenia produkowane przez firmę AEG, jak i urządzenia oferowane przez innych wytwórców: IBM, DEC, Intel, Motorola. Jako medium transmisyjne może służyć skrętka, kable koncentryczne, lub światłowody, przy czym wymagania co do medium są określone dla każdego poziomu indywidualnie [1].

Net/One firmy Ungermann – Bass Inc.

Net/One został opracowany z myślą o łączeniu w sieć urządzeń pochodzących od różnych producentów. Jest to sieć o topologii magistralowej. Połączenie urządzenia z siecią zapewnia tzw. procesor sieciowy, wyposażony w nadajnik-odbiornik komunikacyjny oraz interfejsy szeregowe RS232C (w liczbie 4) oraz równoległe TTL (2), za których pośrednictwem urządzenia są sprzęgane z procesorem sieciowym. Jako medium transmisyjne służy kabel koncentryczny. Pojedynczy segment sieci może osiągać długość ok. 1200 m, a zwiększenie zasięgu transmisji wymaga użycia tzw. powtarzaczy. W jednym segmencie może znajdować się do 250 procesorów sieciowych. Maksymalna prędkość transmisji na kablu głównym – 4 Mb/s, na magistrali szeregowej z interfejsem RS232C – 19 200 b/s [5].

Sieć Net/One nie jest siecią przeznaczoną wyłącznie do zastosowań przemysłowych. Jest również wykorzystywana do automatyzacji prac biurowych i eksperymentów naukowych [2].

Bitbus firmy Intel Corp.

System ten pojawił się na rynku z chwilą, gdy w rozproszonych systemach sterowania zaczęto stosować mikrokomputery jednoukładowe. Umożliwia on połączenie w sieć do 250 urządzeń kablem płaskim bądź skrętką. Zasięg transmisji waha się od 30 m dla transmisji synchronicznej, do kilku tysięcy metrów dla transmisji asynchronicznej. Maksymalna prędkość transmisji – 2,4 Mb/s. Wymiana danych odbywa się według protokołu HDLC/SDLC. Dla realizacji systemów sterowania na podstawie mikrokomputerów jednoukładowych z rodziny MCS51 firma Intel wyprodukowała mikrosterownik 8044 zawierający w jednej strukturze CPU 8051 i sterownik sieciowy. Mikrosterownik ten stanowi również bazę kart rozszerzających dla popularnych mikrokomputerów osobistych klasy IBM PC, co pozwala na ich zastosowanie w tych systemach [14].

Inteldigit-Proway konstrukcji PIAP

Jest to przemysłowa sieć lokalna typu magistralowego, skonstruowana przez Przemysłowy Instytut Automatyki i Pomiarów w Warszawie. Jako medium transmisyjne wykorzystano kable koncentryczne. Na wspólnej magistrali, o maksymalnej długości 2 km, może pracować do 100 urządzeń. Maksymalna prędkość transmisji wynosi 1 Mb/s. Protokół transmisji – Proway (*Process Data Highway*), opracowany przez komitet IEC/TC65 i przyjęty do zbioru standardów [18].

Wśród wyżej przedstawionych systemów przemysłowych przeważają systemy o konfiguracji magistralowej. Jako przyczynę tak szerokiego stosowania tej konfiguracji podaje się [16]:
- prostotę budowy sieci,
- dużą niezawodność – uszkodzenia jednego z węzłów nie powoduje zakłóceń w pracy pozostałych.

Najczęściej stosowanym medium transmisyjnym są kable koncentryczne. Zastosowanie światłowodów zostało ograniczone do systemów, w których występują połączenia typu punkt-punkt (np. sieci pierścieniowe firmy Bailey Controls). W większości przypadków, prędkość przesyłu danych jest rzędu megabitów na sekundę.

W dalszej części artykułu zostanie dokonane porównanie różnych systemów. Informacje na ich temat zebrano w postaci tabeli, w której pamiętano systemy Net/One oraz Bitbus ze względu na to, że są one dobrze scharakteryzowane w artykułach specjalistycznych [5, 14]. Tabelę tę rozszerzono o sieć Arcnet, która jest ciekawym rozwiązaniem z punktu widzenia aplikacji przemysłowych [9, 15, 19].

Środki łączności stosowane w systemach przemysłowych

Na środki łączności w systemach przemysłowych składają się media transmisyjne, protokoły oraz metody dostępu do medium transmisyjnego.

Media transmisyjne

Medium transmisyjne to fizyczny nośnik informacji mający zadanie zapewnienia przesyłu danych pomiędzy poszczególnymi komponentami systemu. W rozwiązaniach systemów przemysłowych dominują media przewodowe (najczęściej metalowe). Charakterystykę tych mediów przedstawiono w tabeli 1. Bardziej szczegółowe informacje na ich temat można znaleźć w pracy [19].

Tabela 1. Porównanie mediów transmisyjnych

	Kabel płaski	Skrętka		Kable koncentryczne		Światłowód
		klasyczna	ekranowana	typu baseband	typu broadband	
Pasmo przenoszenia	rzędu MHz	rzędu MHz[1]	rzędu MHz	rzędu dziesiątek MHz	rzędu setek MHz	rzędu GHz
Zakres stosowalności	łącze dwupunktowe[2]	łącze dwupunktowe[2]	łącze dwupunktowe[2]	łącze wielopunktowe	łącze wielopunktowe	łącze dwupunktowe[3]
Szybkość transmisji w Mb/s	< 2	< 2	< 2[4]	< 12	< 50	< 560
Tłumienność	zależna od częstotliwości	zależna od częstotliwości	zależna od częstotliwości	zależna od częstotliwości	zależna od częstotliwości	niezależna od częstotliwości
Odporność na zakłócenia	mała	mała	średnia	duża	duża	bardzo duża
Poziom emisji zakłóceń	duży	duży	mały	mały	mały	nie emituje
Przechwycenie informacji przez osoby niepowołane	łatwe[5]	łatwe[5]	utrudnione[6]	łatwe[7]	łatwe[7]	bardzo trudne
Ciężar (na km)	mały	mały	mały	duży[8]	duży[8]	mały

[1] Skrętki telefoniczne są wykonywane jako wiązki par przewodów; w celu uniknięcia przesłuchów na sąsiednie pary przewodów pasmo przenoszenia zostało ograniczone do 3000 Hz. Pasmo to pozwala na osiągnięcie prędkości transmisji do 4800 b/s w łączach komutowanych i 10 000 b/s w łączach trwałych.

[2] Możliwe jest tworzenie łącz wielopunktowych, lecz wymaga to fizycznego naruszenia ciągłości medium transmisyjnego, a co za tym idzie, przerwania pracy sieci; ponadto nowa stacja stanowi potencjalne źródło zakłóceń.

[3] Złącza i przyłącza wnoszą tłumienność porównywalną z tłumiennością światłowodu, stąd możliwości stworzenia łącza wielopunktowego są ograniczone do kilku stacji.

[4] W zmodyfikowanej sieci IBM Token Ring przewiduje się zwiększenie prędkości transmisji do 16 Mb/s.

[5] Przez analizę emitowanego pola magnetycznego.

[6] Użycie ekranu zmniejsza poziom emitowanego pola magnetycznego.

[7] Przez użycie przyłącz penetrujących. Włączenie nowej stacji nie wymaga wstrzymywania pracy sieci.

[8] Dochodzi do setek kilogramów.

Do transmisji danych w systemach pracujących w środowisku o dużym poziomie zakłóceń, takim jak środowisko przemysłowe, najlepiej nadają się światłowody. Światłowody charakteryzuje bardzo duża odporność na zakłócenia elektromagnetyczne, niewrażliwość na różnicę potencjałów w punktach końcowych łącza, są lekkie. Można je prowadzić w sąsiedztwie kabli energetycznych i w pobliżu źródeł pola magnetycznego. W przypadku mechanicznego uszkodzenia łącza światłowodowego nie występuje iskrzenie, co ma niebagatelne znaczenie w przypadku pracy w środowisku grożącym wybuchem. Jednak ze względu na trudności w tworzeniu łączy wielopunktowych, światłowodów używa się najczęściej w łączach dwupunktowych, np. w sieciach pierścieniowych, stanowiących zbiór jednokierunkowych łączy dwupunktowych, czy też do łączenia segmentów sieci rozmieszczonych w różnych budynkach.

Skrętka telefoniczna odznacza się co prawda gorszymi parametrami niż światłowody, lecz wykorzystanie do celów transmisji danych istniejącej w obrębie zakładu przemysłowego linii telefonicznej przynosi znaczne obniżenie kosztu całości instalacji [8]. W zależności od użytych układów nadajników-odbiorników, typu skrętki oraz prędkości transmisji, informację cyfrową można przesyłać na odległość nawet do kilkudziesięciu kilometrów [9]. Użycie modemów także może zwiększyć zasięg transmisji.

Najczęściej stosowanym medium transmisyjnym są kable koncentryczne. Cechuje je mniejsza, niż w przypadku skrętki, wrażliwość na zakłócenia elektromagnetyczne i znacznie wolniejszy wzrost tłumienności w funkcji częstotliwości. Kable te umożliwiają łatwe tworzenie liniowych łączy wielopunktowych. W przypadku kabli o litym przewodzie wewnętrznym i większej średnicy, na rynku są dostępne tzw. przyłącza penetrujące, które pozwalają na dołączenie nowej stacji bez konieczności przecinania kabla, a tym samym wstrzymywania pracy sieci. Natomiast wadą tych kabli jest niekiedy ich znaczny ciężar dochodzący na km, do setek kilogramów (w przypadku grubych kabli).

Protokoły

Protokół komunikacyjny, to formalny zbiór przepisów określających strukturę i sposób wymiany informacji między komponentami systemu rozproszonego [11, 17]. Zadaniem protokołu jest zapewnienie bezbłędnego przesyłu danych (przez detekcję błędów i retransmisję, bądź korekcję błędów) oraz ich przezroczystości informacyjnej, a także kodowanie informacji oraz synchronizacja. Przez przezroczystość informacyjną rozumie się możliwość przesłania dowolnej sekwencji bitów, nawet takiej, która w danym protokole ma znaczenie specjalne, np. flaga początkowa i końcowa ramki w protokole HDLC, znak początku tekstu, czy też zapytanie w protokole BSC. Tabela 2 zawiera porównanie różnych protokołów. Informacje na temat organizacji ramek informacyjnych oraz sposobu wymiany danych przy wykorzystaniu danego protokołu można znaleźć w pracach [11, 16, 17].

Z tabeli 2 wynika, że we wszystkich protokołach jest realizowana jedynie detekcja błędów za pomocą cyklicznej sumy kontrolnej CRC lub kontroli podłużnej LRC. Negatywne potwierdzenie odbioru ramki lub brak takiego potwierdzenia wymusza jej retransmisję. Korekcja błędów, przy przesyłaniu długich ramek, wymagałaby znacznej liczby bitów nadmiarowych, co zmniejszyłoby efektywność protokołu, definiowaną jako stosunek liczby bitów danych do liczby bitów w ramce. Najczęściej używanymi wielomianami generacyjnymi sumy cyklicznej są CRC-16 oraz CCITT [16, 19]. Wielomiany te generują 16-bitową sumę kontrolną pozwalającą na wykrycie błędów pojedynczych i podwójnych. Sprzętowa realizacja generacji sumy wymaga użycia jedynie rejestrów przesuwnych wraz z odpowiednio dobranymi sprzężeniami zwrotnymi [13].

Tabela 2. Porównanie protokołów

	BSC[1]	DDCMP[2]	HDLC[3]	SDLC[4]	Modbus[5]	
					RTU	ASCII
Typ protokołu	znakowy	znakowy	bitowy	bitowy	bitowy	znakowy
Rodzaj łącza	dwupunktowe, wielopunktowe	dwupunktowe, wielopunktowe	dwupunktowe, wielopunktowe	dwupunktowe, wielopunktowe, pętlowe	dwupunktowe, wielopunktowe	dwupunktowe, wielopunktowe
Tryb transmisji	synchroniczna	synchroniczna, asynchroniczna	synchroniczna	synchroniczna	synchroniczna	asynchroniczna
Rodzaj kanału	półdupleks	dupleks, półdupleks	dupleks, półdupleks	dupleks, półdupleks	półdupleks	półdupleks
Sekwencja kontrolna	LRC/CRC	CRC	CRC	CRC	CRC	LRC
Adresowanie	indywidualne	indywidualne	indywidualne, rozgłaszanie	indywidualne, rozgłaszanie	indywidualne, rozgłaszanie	indywidualne, rozgłaszanie

[1] Opracowany przez firmę IBM (wprowadzony w 1966 r.), znany również pod nazwą BISYNC.
[2] Opracowany przez firmę DEC.
[3] Opracowany wspólnie przez ANSI i ECMA, zatwierdzony przez ISO jako standard międzynarodowy.
[4] Opracowany przez firmę IBM.
[5] Opracowany przez firmę Gould Inc.

Metody dostępu do medium transmisyjnego

Metoda dostępu określa sposób, w jaki pojedynczy komponent systemu rozproszonego rozstrzyga, w którym momencie uzyskuje on prawo do „zawłaszczenia" medium transmisyjnego oraz przez jaki czas prawo to posiada. Rozróżnia się trzy grupy metod dostępu [6]:

- przydziałowe,
- kontrolowanego dostępu,
- przypadkowego dostępu.

Istotą metod przydziałowych jest przypisanie poszczególnych zasobów medium (np. pewnego zakresu częstotliwości z pasma przenoszenia medium) wszystkim urządzeniom pracującym w sieci.

W metodach kontrolowanego dostępu urządzenia przekazują sobie wzajemnie tzw. uprawnienie (zezwolenie) na zajęcie medium lub też „wydawaniem" zezwoleń zajmuje się wyróżnione urządzenie (zwane urządzeniem *Master* bądź arbitrem). Urządzenie, które otrzymało uprawnienie, ma całkowitą kontrolę nad łączem fizycznym, a tym samym ma prawo do nadawania informacji. Po pewnym czasie urządzenie to musi przekazać uprawnienie następnemu. Metody kontrolowanego dostępu pozwalają na oszacowanie maksymalnego czasu oczekiwania na dostęp do medium transmisyjnego, przy czym czas ten jest uzależniony od liczby urządzeń w systemie oraz charakterystyk sieci. Ta klasa metod (zwanych także metodami deterministycznymi) obejmuje metody przepytywania oraz metody przekazywania uprawnienia np. **token bus** dla sieci o konfiguracji magistralowej, drzewiastej oraz gwiaździstej i **token ring** dla sieci o konfiguracji pierścieniowej.

W metodach przypadkowego dostępu, zwanych także metodami rywalizacyjnymi, każde urządzenie może rozpocząć nadawanie zaraz po stwierdzeniu, że medium transmisyjne nie jest zajęte, tzn. że nie odbywa się transmisja. Oczywiście może wystąpić sytuacja, że więcej niż jedno urządzenie będzie chciało nadawać w tym samym momencie. Zaistnienie takiej sytuacji, tzw. kolizji, powoduje zniekształcenie nadawanej informacji, a w konsekwencji brak potwierdzenia jej odebrania lub też natychmiastowe przerwanie transmisji. Po pewnym czasie urządzenia biorące udział w kolizji usiłują ponownie „zawłaszczyć" medium. Najbardziej znaną metodą z tej grupy jest CSMA/CD (ang. *Carrier Sense Multiple Access with Collision Detection*). Z racji niederministycznego charakteru tej metody trudno jest

oszacować maksymalny czas oczekiwania na dostęp, ten bowiem zależy od aktualnego obciążenia sieci. Bardziej szczegółowe informacje na temat metod dostępu można znaleźć w pracach [2, 5, 19].

W przemysłowych systemach sterowania, gdzie występuje konieczność przesyłania informacji o stanie procesu (w odstępach czasu określonych jego specyfiką, przy czym odstęp ten nie może być większy od pewnej ustalonej wartości), przeważają deterministyczne metody dostępu. Inne rozwiązanie obrała firma Ungermann-Bass Inc., która w swej sieci Net/One, przeznaczonej także do zastosowań w sferze sterowania, wykorzystała metodę CSMA/CD [2]. Opracowano także odmianę tej metody o nazwie CSMA/CA (ang. *Carrier Sense Multiple Access with Collision Avoidance*) [7]. W tym przypadku, każde urządzenie w sieci ma priorytet. Odmiennie niż w CSMA/CD, gdzie każde urządzenie może rozpocząć transmisję natychmiast po stwierdzeniu ciszy na łączu, urządzenie zamierzające nadawać musi najpierw stwierdzić fakt przejścia linii ze stanu aktywnego w nieaktywny. Po wystąpieniu tego zdarzenia urządzenie zamierzające nadawać odczekuje odstęp czasu proporcjonalny do jego priorytetu. Jeśli więcej niż jedno urządzenie ma informację do przesłania i wszystkie wykryją „tę samą" deaktywację linii, wówczas rywalizację wygrywa urządzenie o najwyższym priorytecie. Wystąpienie transmisji danych w trakcie odczekiwania sprawia, że ubiegające się o dostęp urządzenia rozpoczynają odliczanie czasu od początku. Gdy cisza na łączu trwa zbyt długo, dowolne urządzenie wysyła „pustą" ramkę (ang. *dummy frame*) dla wymuszenia przejścia „linia aktywna-nieaktywna", co jest warunkiem poprawnej pracy sieci.

Wgrupie kontrolowanych metod dostępu ciekawym rozwiązaniem jest odmiana metody przekazywania uprawnienia, opracowana przez firmę Datapoint Corp. i zastosowana w sieci Arcnet, będącej produktem tejże firmy [15]. Była to pierwsza dostępna na rynku lokalna sieć komputerowa. Urządzenie, po otrzymaniu uprawnienia do nadawania, najpierw „upewnia się", czy odbiorca może przyjąć informację. Jeżeli otrzyma odpowiedź negatywną, przekazuje uprawnienie następnemu urządzeniu i po kolejnym otrzymaniu uprawnienia, ponawia próbę. Wstępne sprawdzanie gotowości przyjęcia danych przez odbiorcę pozwala uniknąć transmisji ramek, które i tak zostaną odrzucone ze względu na brak wolnego bufora odbiorczego. Zmniejsza się przez to obciążenie linii transmisyjnej. W sieciach przemysłowych przesyła się najczęściej krótkie ramki o długości kilku bajtów [9]. Sieć Arcnet została zaprojektowana pod kątem szybkiego i niezawodnego przesyłania takich ramek, co predystynuje to rozwiązanie do zastosowań przemysłowych. Dzięki tej odmianie metody przekazywania uprawnienia, mimo małej w porównaniu z innymi sieciami prędkości transmisji (2.5 Mb/s), w sieci Arcnet uzyskuje się dużą efektywność przesyłania danych przy stosunkowo małym koszcie sprzętu i małych narzutach programowych.

Porównanie przemysłowych rozproszonych systemów sterowania

Tabela 3 zawiera porównanie systemów przeznaczonych do zastosowań przemysłowych, rozszerzone o informacje dotyczące sieci Arcnet firmy Datapoint Corp. Sieć ta znalazła się w tej klasyfikacji ze względu na cechujące ją właściwości. Arcnet jest interesującym rozwiązaniem nie tylko z powodu zastosowanej w tym systemie metody dostępu do łącza fizycznego, będącej odmianą metody przekazywania uprawnienia typu *token bus*. Sieć ta sama śledzi zmiany zachodzące w jej otoczeniu (uszkodzenie węzła, wyłączenie węzła, dołączenie nowego węzła) i automatycznie, bez złożonych procedur obsługi takich sytuacji, dostosowuje się do tych zmian [15, 19]. Tabela została opracowana na podstawie materiałów firmowych (systemy Network 90 i INFI-90 firmy Bailey Controls, Modbus firmy Gould Inc., GeΔmatics firmy AEG) oraz dostępne pozycje literaturowe [15, 16, 19]. Ze względu na brak niektórych danych w materiałach firmowych, w tabeli występują puste miejsca.

Tabela 3. Porównanie rozwiązań sieci przemysłowych

	Network 90		INFI-90	Modbus	GeΔmatics	Proway	Arcnet
	Plant Loop	Super Loop					
Konfiguracja	pierścień	pierścień	pierścień	magistrala	magistrala	magistrala	gwiazda, drzewo, magistrala
Medium transmisyjne	kable: płaskie, koncentryczne	kable: płaskie, koncentryczne	kable: płaskie, koncentryczne, światłowody	skrętka ekranowana, kable koncentryczne, kable telefoniczne, środki łączności radiowej i mikrofalowej	skrętka, kable koncentryczne, światłowody (patrz 1)	kable koncentryczne	skrętka telefoniczna, kable koncentryczne, światłowody, środki transmisji bezprzewodowej
Protokół transmisji				Modbus--RTU, Modbus--ASCII	patrz 2	HDLC, IEC/TC57, IEEE802[3]	patrz 4
Maksymalna liczba urządzeń	63	250[5]	250[6]	32[7]		100	19[8]
Maksymalna odległość między urządzeniami	2000m	2000m$_{2Mb/s}$, 1000m$_{10Mb/s}$ (patrz 8)	patrz 9	4752m[10]	patrz 11	2000m[12]	660m[13]
Metoda dostępu do łącza				nadawanie po otrzymaniu zezwolenia od jednostki MASTER	patrz 14	przekazywanie uprawnienia	przekazywanie uprawnienia
Maksymalna prędkość transmisji	500 kb/s	10 Mb/s	10 Mb/s	19.2 kb/s	patrz 15	1 Mb/s	2.5 Mb/s

[1] System komunikacyjny GeΔmatics-a został podzielony na trzy poziomy:
MODNET 1 – poziom interfejsu z obiektem (ang. *field communication*),
MODNET 2 – poziom komunikacji z procesem (ang. *process communication*),
MODNET 3 – poziom nadzoru pracy poszczególnych obiektów oraz sprzęgnięcia z sieciami rozległymi i publicznymi (ang. *bockbone network*).
Wymagania co do medium transmisyjnego są określone indywidualnie dla każdego z poziomów:
MODNET 1 – skrętka, światłowody,
MODNET 2 – kable koncentryczne, światłowody,
MODNET 3 – kable koncentryczne szerokopasmowe, światłowody.

[2] Na różnych poziomach systemu komunikacyjnego GeΔmatics-a używa się różnych protokołów:
MODNET 3 – SDLC,
MODNET 2 – MAP 3.0 zgodny z modelem odniesienia ISO/OSI na poziomie trzech warstw,
MODNET 3 – MAP 3.0 w pełni zgodny z modelem odniesienia oraz TOP 3.0.

[3] Protokół HDLC jest stosowany w sieci Proway A, protokół IEC/TC52 w sieci Proway B, natomiast IEEE802 – w sieci Proway C.

[4] Formaty ramek oraz sposób współpracy urządzeń w sieci są autorstwa firmy Datapoint Corp.

[5] Jako urządzenie pracujące w sieci Super Loop traktuje się również podsieć tego samego typu. Stąd w całym systemie może pracować do 250 × 250 = 62 500 urządzeń.

[6] Również sieć typu INFI-90 jest traktowana jako urządzenie w sieci INFI-90. Liczba urządzeń w całym systemie może dochodzić do 62 500.

[7] Liczba ta odnosi się do systemów zbudowanych na podstawie modemów produkowanych przez firmę Gould Inc. W przypadku użycia modemów komercyjnych liczba urządzeń wzrasta do 247.

[8] Przy długości kabla głównego (współosiowego) równej 60 m, przy kilku stacjach dopuszcza się stosowanie kabla o długości do 300 m.

[9] Dane te dotyczą systemów, w których jako medium transmisyjne wykorzystuje się kable płaskie. Przy użyciu kabli koncentrycznych odległości te zostają podwojone.

[10] W tabeli podano długość kabla głównego (skrętka ekranowana, 32 urządzenia SLAVE, modemy firmy Gould). Dopuszczalne jest użycie kabli dystansowych o długości nie przekraczającej 150 m. W przypadku użycia modemów komercyjnych odległość ta nie jest limitowana. Stosując środki łączności radiowej, wartość ta wynosi odpowiednio (w terenie płaskim):
system UHF 30 W – 14 km,
system VHF 5 W – 3,2 km.

[11] MODNET 1 – 1,3 km,
MODNET 2 – 1 km,

MODNET 3 – brak danych.

[12] W tabeli podano długość kabla głównego. Dopuszczalne jest użycie kabli dystansowych o długości nie przekraczającej 100 m.

[13] Segmenty Arcnet-a można sprzęgać odcinkami światłowodowymi o długości dochodzącej do 1200 m.

[14] Metody dostępu do łącza stosowane na poszczególnych poziomach:
MODNET 1 – po otrzymaniu zezwolenia na nadawanie od urządzenia MASTER,
MODNET 2 – przekazywanie uprawnienia,
MODNET 3 – przekazywanie uprawnienia, CSMA/CD.

[15] Maksymalne prędkości transmisji na poszczególnych poziomach:
MODNET 1 – 375 kb/s,
MODNET 2 – 5 Mb/s,
MODNET 3 – 10 Mb/s.

W większości rozwiązań przyjęto zasadę zdecentralizowanego sterowania dostępem do medium transmisyjnego z wykorzystaniem metod deterministycznych. W systemie Modbus firmy Gould Inc. [10] zastosowano podejście odmienne, w którym zarządzaniem dostępem do łącza zajmuje się urządzenie Master. Ze względu na to, że w dostępnej literaturze można znaleźć informacje na temat protokołów objętych normami międzynarodowymi (np. HDLC [11, 16, 17, 19]), Proway [16], czy też będących produktami znanych szerokiemu kręgowi użytkowników firm światowych (np. DDCMP firmy DEC [16], SDLC [11, 17], Arcnet [15, 19], w artykule zostanie przedstawiony jedynie protokoł zastosowany w systemie Modbus.

Istnieją dwie odmiany protokołu Modbus:
- Modbus-ASCII,
- Modbus-RTU (ang. *Remote Terminal Unit*).

Adres odbiorcy	Kod funkcji	Dane	Sekwencja kontrolna

Rys. 2. Format ramki informacyjnej protokołu Modbus

Różnice między obu odmianami protokołu przedstawia tabela 4. Informacje podane w tej tabeli są rozszerzeniem informacji z tabeli 2. Format ramki tego protokółu przedstawia rys. 2.

Tabela 4. Protokoły Modbus-ASCII i Modbus-RTU – różnice

	Modbus-ASCII	Modbus-RTU
Tryb transmisji	asynchroniczna	synchroniczna
System kodowania	szesnastkowy - znaki ASCII 0-9, A-F	binarny
Struktura znaku: – bit startu – bity danych – bit parzystości (opcjonalnie) – bity stopu	 1 7 1 1 lub 2	 1 8 1 1 lub 2
ograniczniki ramki: – początek ramki – koniec ramki	znak ASCII ":" znaki ASCII "LF" & "CR" lub cisza na łączu równa 1 s.	znak otrzymany po okresie ciszy na łączu równej czasowi transmisji 3 znaków cisza na łączu równa czasowi transmisji 3 znaków.
Sekwencja kontrolna	LRC	CRC

Protokół Modbus umożliwia komunikację w dwóch trybach:
- między urządzeniem nadrzędnym Master a wybranym urządzeniem podrzędnym Slave. Wybór urządzenia Slave odbywa się przez jego zaadresowanie liczbą z zakresu 1-247. W terminologii używanej w materiałach firmowych Gould Inc. ten rodzaj komunikacji jest nazywany komunikacją typu „pytanie-odpowiedź",
- rozgłaszania, w którym urządzenie Master rozsyła informację jednocześnie do wszystkich urządzeń Slave. Adres 0 stanowi adres rozgłaszania.

Kod funkcji określa rodzaj operacji, którą ma wykonać urządzenie podrzędne, np. odczyt stanu wejść cyfrowych, odczyt stanu wyjść cyfrowych, zmiana stanu itp. W polu danych znajdują się informacje niezbędne do realizacji zleconej operacji, np. numer wyjścia, którego stan ma ulec zmianie. Urządzenie Slave, po wykonaniu zlecenia, przesyła do urządzenia Master odpowiedź, podając własny adres w polu „Adres odbiorcy". Stanowi to jednocześnie potwierdzenie poprawnego odbioru ramki przez urządzenie Slave. Odbiór informacji rozsyłanej w trybie rozgłaszania nie jest potwierdzany.

Wybór protokołu jest uzależniony od tego, czy urządzenia w systemie są przystosowane do realizacji transmisji synchronicznej, czy asynchronicznej (patrz tabela 2).

W przypadku pozycji „Metoda dostępu do łącza" dla produktów firmy Bailey Controls można wziąć pod uwagę dwie metody [19]:
1. przekazywania uprawnienia (wersja *token ring*),
2. wirujących tacek (ang.. *slotted ring*).

Istota metody przekazywania uprawnienia została już przybliżona w rozdziale „Metody dostępu do medium transmisyjnego". W wersji *token ring* „wolna" ramka uprawnienia krąży między urządzeniami. Gdy urządzenie, które chce nadawać informację otrzyma uprawnienie, zaznacza je jako „zajęte" i przekształca w ramkę informacyjną. Ramka informacyjna po pełnym obiegu pierścienia zostaje z niego usunięta. W drugiej z wymienionych metod, w pierścieniu krąży kilka tzw. tacek, mogących pomieścić informację ograniczoną do kilkudziesięciu bitów. Informację do przesłania dzieli się na ciąg ramek o długości nie przekraczającej pojemności tacki i wysyła kolejno, zajmując nadchodzące puste tacki. Zadaniem zwalniania tacek jest obarczony nadawca. Ponieważ w jednej chwili w pierścieniu mogą krążyć tacki wypełnione informacją od różnych nadawców, a adresowanych do tego samego odbiorcy, oprogramowanie komunikacyjne musi być uzupełnione o mechanizmy segregacji ramek na podstawie adresu nadawcy.

W literaturze podaje się jeszcze jedną metodę dostępu do łącza w sieciach pierścieniowych, nazywaną metodą wstawianych rejestrów (ang. *register insertion ring*). Dla realizacji tej metody układ nadajnika-odbiornika musi być wyposażony w dwa rejestry przesuwające, o długości nie mniejszej niż maksymalna dopuszczalna długość ramki w sieci. W jednym z rejestrów kompletuje się ramkę przeznaczoną do wysłania. W drugim rejestrze jest gromadzona informacja odbierana. Rejestr ten pełni także rolę retransmitera. Przy braku odbioru i wystarczającej ilości miejsca w rejestrze nadawczo-odbiorczym, przygotowaną uprzednio ramkę można do niego przepisać i rozpocząć nadawanie. Pierwsze systemy eksperymentalne, w których zastosowano tę metodę dostępu, powstały już w końcu lat siedemdziesiątych, do dnia dzisiejszego jednak nie wprowadzono takich systemów na rynek.

* * *

W artykule opisano środki łączności stosowane w przemysłowych rozproszonych systemach sterowania. Struktury rozproszone są obecnie dominującą grupą w tej klasie systemów. Złożyły się na to następujące przyczyny:
- konieczność przetwarzania dużej ilości informacji o sterowanym procesie technologicznym w czasie rzeczywistym,
- konieczność szybkiej reakcji na sytuacje alarmowe,
- rozległość (w sensie terytorialnym) automatyzowanych obiektów,
- hierarchiczna struktura systemu sterowania,
- konieczność zapewnienia dużej niezawodności i dyspozycyjności systemu.

Zakres stosowania scentralizowanych systemów sterowania ogranicza się do obiektów małych, przy mniejszych wymaganiach dotyczących czasu reakcji [18].

Pojęcie środków łączności obejmuje zarówno media transmisyjne, jak też protokoły i metody dostępu do łącza fizycznego. Ponieważ literatura na temat organizacji transmisji danych i lokalnych sieci komputerowych jest szeroko dostępna, w artykule podano jedynie podstawowe wiadomości dotyczące tych trzech elementów, składających się na pojęcie środków łączności. Wiadomości te rozszerzono przy omawianiu rozwiązań dotychczas nie prezentowanych w piśmiennictwie, spośród których najwięcej uwagi poświęcono produktowi firmy Gould Inc., systemowi Modbus. Zostało to podyktowane tym, że w materiałach tej firmy zamieszczono wszystkie interesujące informacje (patrz tab. 3). W tabeli pominięto systemy: Net/One, Bitbus i Inteldigit-Proway. Dwa pierwsze są dość dobrze przedstawione przez swoich twórców w artykułach specjalistycznych: Net/One w [5], Bitbus w [14]. Parametry krajowego rozwiązania, systemu Inteldigit-Proway, są zgodne z normą Proway [18]. Arcnet znalazł się w tej tabeli ze względu na zespół cech czyniących tę sieć interesującym rozwiązaniem z punktu widzenia zastosowań przemysłowych. Są to:

- wyeliminowanie transmisji tych ramek, które zostałyby odrzucone z powodu braku wolnego bufora u odbiorcy; zmniejsza to obciążenie linii transmisyjnej;
- deterministyczna metoda dostępu do łącza fizycznego (przekazywanie uprawnienia), pozwalająca na łatwe oszacowanie maksymalnego czasu oczekiwania na dostęp;
- automatyczna rekonfiguracja pierścienia logicznego (pod tym pojęciem rozumiemy kolejność obiegu uprawnienia), dokonywana przez węzły sieci po wystąpieniu zmian w konfiguracji, np. wyłączeniu węzła, dołączeniu nowego węzła;
- automatyczne odtworzenie pierścienia logicznego po zagubieniu uprawnienia.

Należy również wziąć pod uwagę to, że reakcja na zmianę w konfiguracji lub zagubienie uprawnienia nie wymaga stosowania złożonych procedur obsługi takich zdarzeń. Od końca lat osiemdziesiątych są prowadzone prace mające na celu wprowadzenie tego rozwiązania do zbioru standardów.

Z tabeli 3 wynika również, że w systemach przemysłowych:

- przeważają systemy o organizacji magistralowej;
- najczęściej używanym medium transmisyjnym są kable koncentryczne; przyczyną częstego stosowania tego typu kabli jest, oprócz dużej odporności na zakłócenia, łatwość tworzenia łączy liniowych wielopunktowych;
- w większości systemów przyjęto zasadę zdecentralizowanego sterowania dostępem do medium transmisyjnego z użyciem metod deterministycznych.

W niektórych rozwiązaniach dopuszcza się użycie kabli telefonicznych. Umożliwia to wykorzystanie do celów transmisji danych już istniejącej w zakładzie przemysłowym sieci telefonicznej. Obniża to całkowity koszt systemu, bowiem koszt okablowania i jego instalacji stanowi znaczącą pozycję.

LITERATURA

[1] AEG GeAmatics. The EAG Automation System
[2] Allan R.: Local-net architecture, protocol issues heating up. Electronic Design, April 16, 1981
[3] Bailey Controls Bailey INFI-90. Strategic Process Management Systems
[4] Bailey Controls Network 90. Distributed Process Controls
[5] Bass Ch., Kennedy J. S., Davidson J. M.: Local network gives flexibility to distributed processing. Electronics, September 25, 1980
[6] Bojdak A., Kołodziej J., Kossowski S., Rozmus M.: Sieci lokalne dla systemów mikrokomputerowych". Materiały szkoleniowe nr 1, 1988
[7] Court R.: Real-time Ethernet. Computer communications, Vol 15, No 3, April 1992
[8] Gawlowicz D. J.: Use Existing In-Plant Telephone Lines for Data Tranmission. Control Engineering, May 1978
[9] Gingold D., Veal B. L.: Arcnet and NetBios: Low-Cost Communication Combination. Control Engineering, October 1990
[10] Gould Inc. Gould Modbus Industrial Communication System, 1981
[11] Guilbert J. F., Macchi C.: Tele-informatics. Data and Computer Communications, North-Holland, 1985
[12] Güth R.: Computer Systems For Process Control. Plenum Press, 1986
[13] Hedtke R.: Systemy mikroprocesorowe. WNT, 1987
[14] Mac Williams P., Wolochow P. I., Zasloff R. H.: Microcontroller serial bus yields distributed multispeed control. Electronics, February 9, 1984
[15] Murphy J. A.: Token-passing protocol boosts throughput in local networks, Electronics, September 8, 1982
[16] Orłowski H.: Komputerowe układy automatyki. WNT, 1987
[17] Osgood G. S., Sliger E. J.: Communications Protocol is Aimed at Low-Cost Remote I/O Systems. Control Engineering, June, 1980
[18] Plaza R. A., Wróbel E. J.: Systemy czasu rzeczywistego. WNT, 1988
[19] Wolisz A.: Podstawy lokalnych sieci komputerowych. T. 1. Sprzęt sieciowy. WNT, 1990.

CeBIT'93

dokończenie ze s. 12

- „Centrum badań strategicznych" (*Strategic Research Center*) – w hali nr 22;
- „Biznes przez satelitę" (*Satellite Business*) – hali nr 23;
- „Szanse roku 2000" (*Chancen* 2000) – w hali nr 28.

Przez cały czas otwarcia Targów (godziny od 9 do 18) w licznych audytoriach Targowego Centrum Kongresowego odbędzie się osiem dużych konferencji tematycznych i branżowych oraz ponad 200 prezentacji osiągnięć czołowych wystawców, z którymi będzie mogło zapoznać się ponad 10 000 osób.

Jedną z bardziej interesujących inicjatyw organizatorów CeBIT jest zapoczątkowana w 1989 r. akcja prezentowania tzw. krajów partnerskich, skłaniająca wybrane kraje do bardziej aktywnego uczestnictwa w Targach. Dotąd były to następujące kraje: Indie (1989), USA (1990), Hongkong (1991) oraz Finlandia i Norwegia (1992). W roku bieżącym uznano, że w obliczu pełnej integracji krajów EWG partnerem CeBIT nie będzie już jeden kraj, ale cały region gospodarczy Europy Środkowo-Wschodniej, któremu należy dopomóc w naturalnych dążeniach integracyjnych. Cel ten będzie realizowany przez wspomniane już ekspozycje specjalne oraz Forum w hali nr 6. O wysiłkach organizatorów na tym polu świadczyć może m.in. fakt, że po wielu latach totalnej absencji naszego kraju (przerwanym w ub.r. eksperymentalnym wystąpieniem czterech firm zaproszonych) Polskę w tym roku będzie reprezentować 13 wystawców, co wprawdzie nie imponuje w zestawieniu z liczbą wystawców Taiwanu (249), ale stawia nas na pierwszym miejscu w regionie.

Jako wielokrotny bezpośredni uczestnik różnych imprez międzynarodowych muszę bezstronnie potwierdzić bezkonkurencyjność CeBITu, co w połączeniu ze stosunkową bliskością Hanoweru powinno skłaniać coraz większą liczbę zainteresowanych, i to zarówno profesjonalistów, jak i użytkowników informatyki, do systematycznego odwiedzania tej imprezy. Szczegółowe informacje na temat CeBIT oraz innych imprez organizowanych przez Zarząd Targów Hanowerskich można uzyskać od ich przedstawiciela **p. mgr Zdzisława Gepperta, ul. Mokotowska 48, 00-542 Warszawa, tel. 660-05-05 lub 660-05-25.**

WK

Inteligentne bazy danych

W znanym wydawnictwie amerykańskim John Wiley ukazała się w 1989 r. interesująca książka poświęcona inteligentnym bazom danych*). Książka jest napisana jako typowy podręcznik amerykański, tzn. bardzo przystępnie oraz ze znaczną liczbą przykładów. Stanowi ona materiał aktualny i nowoczesny, prezentując oryginalny dorobek autorów z kilku ostatnich lat..

Na tle przeglądu stanu obecnego przedstawiono zarys przyszłego rozwoju nowych technologii w dziedzinie inteligentnych baz danych. Z treści książki wyłania się bardzo spójna wizja eksplozji informatyki w bieżącej dekadzie.

Książka ma 479 stron tekstu zilustrowanego 180 rysunkami. Jej zawartość w najbardziej syntetycznym ujęciu charakteryzują nazwy poszczególnych rozdziałów: Systemy baz danych; Programowanie obiektowe; Systemy eksperckie; Hipermedia; Zarządzanie tekstem i wyszukiwania informacji; Inteligentne bazy danych.
Szczególnie obszerna jest zamieszczona literatura obejmująca 450 pozycji bibliograficznych, z których przytłaczająca większość pochodzi z lat 1983–1988.

Książkę można czytać wielopłaszczyznowo dobierając spójne fragmenty do poziomu zainteresowań. W celu ogólnego zorientowania się w przedmiocie wystarczy zapoznać się z rozdziałami 1 i 8.. Uzupełniające przeczytanie wstępów i podsumowań do poszczególnych rozdziałów umożliwia dalsze rozszerzanie wiadomości. Osoby interesujące się bardziej szczegółowo określoną tematyką sięgają do wnikliwego studiowania wybranych rozdziałów. Taki sposób prezentacji materiału stanowi istotny walor publikacji napisanej nowoczesnym, komunikatywnym językiem.

Inteligentne bazy danych mogą być ogólnie określone jako bazy danych które zarządzają informacją w sposób naturalny, czyniąc ją łatwą do gromadzenia, udostępnienia i wykorzystania.

Przewaga inteligentnych baz danych nad konwencjonalnymi bazami polega na łatwym i naturalnym ich stosowaniu, możliwości gromadzenia dużych ilości informacji w sposób ciągły oraz zapewnieniu użytkownikowi realizacji jego zadań za pomocą odpowiednich narzędzi zarządzania informacją.

Architektura najbardziej rozwiniętych inteligentnych baz danych składa się z trzech poziomów:
- narzędzia wysokiego poziomu – są to techniki zarządzania informacją łącznie z programowaniem obiektowym,
- sprzęgi wysokiego poziomu, zapewniające pełne współdziałanie z użytkownikiem,
- mechanizm wnioskowania – dysponuje informacją ukierunkowaną obiektowo i zawiera procedury wnioskowania uprzedzającego oraz wstecznego.

Wobec oryginalnego i interesującego ujęcia tematyki – treść poszczególnych rozdziałów publikacji zostanie omówiona nieco szerzej niż w konwencjonalnych recenzjach.

*) K. Parsaye, M. Chignell, S. Knoshafian, H. Wong: „Inteligent Databases".

Systemy baz danych

W początkowych fazach rozwoju informatyki dane zestawiano w odpowiednie układy hierarchiczne. W latach sześćdziesiątych rozpowszechniły się duże systemy komputerowe, rejestrujące dane na dyskach i obsługujące indywidualne potrzeby informacyjne przedsiębiorstw. Wkrótce okazało się, że znaczna część informacji jest wspólna dla różnych zastosowań i dlatego jest celowe dzielenie się dostępem do takich danych. Z upływem czasu w scentralizowanych bazach danych było jednak coraz trudniej tworzyć ich struktury i organizować do nich łatwy dostęp.

Obecnie zalewa nas coraz większy nadmiar informacji, bowiem zdolność ich gromadzenia znacznie przekracza zdolność efektywnego wyszukiwania potrzebnych danych. W bazach danych wprowadzono więc modele sieciowe i hierarchiczne jako uzupełnienie modelu relacyjnego. Zdaniem autorów książki model relacyjny w połączeniu z programowaniem obiektowym będzie najbardziej użytecznym narzędziem przy budowie inteligentnych baz danych. Model relacyjnych baz danych zapewnia prostą strukturę i daje się łatwo powiązać z teorią zbiorów i systemami logicznymi. Wyszukiwanie danych w modelu relacyjnym jest realizowane przez zastosowanie algebry relacji do manipulowania zależnościami. Ostatnio do tego modelu wprowadza się dalsze modyfikacje przez wykorzystywanie standardowych języków SQL (*Structured Query Language*) oraz QBE (*Query-By-Example*). Idea pytań za pomocą przykładu jest stosowana zamiast opisywania procedury zmierzającej do uzyskania potrzebnej informacji. Użytkownik podaje przykłady tego, czego wymaga.

Programowanie obiektowe

Jednym z ewolucyjnych kroków w rozwoju technologii informatycznej było wyraźne odseparowanie programów od danych. Wykonanie tradycyjnych programów było zwykle procesem sekwencyjnym, natomiast program obiektowy nosi charakter systemu dynamicznie generowanych współbieżnych działań. Każdy obiekt stanowi zamkniętą całość, która obejmuje lokalną pamięć oraz operacje jej przyporządkowane. Autorzy wyraźnie podkreślają, że rozwiązanie ukierunkowane obiektowo uzupełnia podejście relacyjne oraz to, że prawidłowo określone systemy ukierunkowane obiektowo mogą być integrowane z konwencjonalnymi bazami danych.

Podstawową cechą inteligentnej bazy danych jest abstrakcyjny model danych zawartych w tej bazie. Pozwala to bezpośrednio zastąpić model użytkownika informacją rzeczywistą. Do trzech zastosowań ukierunkowania obiektowego, jako uzupełnienia reprezentacji wiedzy należą: symulacja zdarzeń dyskretnych, projektowanie inżynierskie oraz inżynieria oprogramowania, która ostatnio wyodrębniła się w samodzielną dyscyplinę.

Systemy eksperckie

Systemy eksperckie stanowią podklasę systemów doradczo-decyzyjnych i są stosowane w przypadku pojawienia się

problemów, które nie nadają się do rozwiązywania metodami czysto algorytmicznymi. Są to programy komputerowe, wykorzystujące zgromadzoną wiedzę oraz mechanizmy wnioskowania do rozwiązania zadań, które zwykle są dokonywane przez ekspertów. Architektura systemów eksperckich, oprócz bazy wiedzy i interpretatora reguł z modułem objaśniającym, obejmuje również sprzęg (interfejs) między systemem i użytkownikiem.

Narzędzia do budowy reprezentacji wiedzy i programów wnioskowania w inteligentnych bazach danych zostały zrealizowane w technice systemów eksperckich. Systemy te obejmują dwa kierunki wspierania baz danych: umożliwiają programowanie wyższego poziomu oraz tworzą formalizm reprezentacji wiedzy. W inteligentnych bazach danych wykorzystuje się zarówno metody wnioskowania wyprzedzającego, jak i wstecznego. W przypadkach bardziej złożonych, w których nie wystarcza zastosowanie logiki dwuwartościowej, wprowadza się mechanizmy wnioskowania w logice rozmytej.

W celu zintegrowania systemów eksperckich z systemami baz danych, stosuje się dodatkowe sprzęgi programowe ułatwiające użytkownikowi bezpośrednią manipulację tymi danymi. Podstawowe moduły systemów eksperckich są przydatne także do konstruowania systemów hipertekstowych oraz do budowy zaawansowanych systemów interpretacji tekstu i efektywnego wyszukiwania informacji, co jest również zadaniem inteligentnych baz danych.

Hipermedia

Systemy hipermediów są w porównaniu z tradycyjnymi metodami liniowej strukturalizacji informacji bardziej elastyczne w gromadzeniu i wyszukiwaniu wiedzy. W tym celu jest wykorzystywane nowe narzędzie – hipertekst, umożliwiający szybki, bezpośredni, bardzo zróżnicowany dostęp do wybranych dokumentów lub ich fragmentów. Hipertekst pozwala łączyć oficjalnie publikowane dokumenty z osobistymi notatkami, uwagami i zdjęciami. W ten sposób tworzy się zindywidualizowaną encyklopedię wiedzy, dostosowaną do potrzeb konkretnego użytkownika.

Z punktu widzenia inteligentnych baz danych, hipertekst stanowi jednocześnie metodę gromadzenia danych i sposób ich wyszukiwania. Stosowany w hipertekście zbiór węzłów i połączeń między nimi pozwala zwielokrotnić wzajemne skojarzenia w reprezentacji wiedzy. Hipermedia zmierzają do zdynaminizowania baz danych i wprowadzenia do nich mechanizmów samomodyfikacyjnych. Ułatwia to selekcję informacji zgodnie z życzeniem użytkownika.

Zarządzanie tekstem i wyszukiwanie informacji

Tekst jest najważniejszym komponentem technologii inteligentnych baz danych. Większość ludzkiej wiedzy nagromadzono w formie tekstu. Należy teraz związać wiedzę tekstu podstawowego z reprezentacją wiedzy podstawowej, którą można wykorzystać do wnioskowania. Chodzi tu nie tylko o stworzenie warunków do natychmiastowego wyszukiwania potrzebnej informacji, ale wykreowanie procesów przekształcających informację tekstową w reprezentację wiedzy, która mogłaby być zrozumiała i wykorzystywana przez system wnioskowania maszynowego.

Wyszukiwanie informacji wiąże się nierozerwalnie z technikami jej reprezentacji, gromadzenia, zarządzania i dostępu. Obecne metody gromadzenia informacji znacznie przekraczają możliwości jej efektywnego zorganizowania i wyszukiwania. Powstały specjalistyczne podprogramy do formułowania pytań w językach baz danych, co pozwala użytkownikowi realizować własne potrzeby informacyjne. Wiedza jest przedstawiana jako zbiór indeksów, które mogą być kojarzone z pytaniami. Stosuje się mechanizmy specjalnych sprzężeń zwrotnych w celu uzyskania bardziej wnikliwej odpowiedzi na postawione pytanie. Są czynione również próby wprowadzenia kryteriów oceny wiarygodności informacji, np. w postaci liczby aktualnie wykorzystywanych przez użytkownika dokumentów.

Podstawowym problemem w zarządzaniu tekstem jest budowa takiego systemu indeksacji, który odpowiadałby potrzebom potencjalnego użytkownika. Autorzy proponują, aby struktura indeksacji opierała się na koncepcji hierarchicznej, która może być zrealizowana w różny sposób. Indeksowanie w bazach danych może być wykonywane ręcznie lub automatycznie. Indeksowanie automatyczne pozwala zwiększyć gęstość indeksacji, sprzyjającej precyzji wyszukiwania potrzebnych fragmentów wiedzy. Autorzy nie widzą istotnej różnicy między tekstowymi i innymi typami baz danych, aczkolwiek w przeszłości problemy te traktowano jako oddzielne. Sposób indeksowania bazy wiedzy jest sprawą krytyczną dla efektywnego funkcjonowania inteligentnych baz danych.

Inteligentne bazy danych

W poprzednich rozdziałach sformalizowano wymagania dotyczące inteligentnych baz danych. Powstał uporządkowany system wiążący końcówki użytkownika ze zunifikowaną strukturą zawierającą hipermedia, języki dialogowe (np. SQL i QBE), programowanie ukierunkowane obiektowo oraz procedury wnioskowania. Zaproponowany przez autorów książki model inteligentnej bazy danych pokazuje, jak w stosunkowo łatwy sposób korzystać z dostępu do baz zawierających nadmiar informacji. Istotną cechą tego modelu jest równoległość czerpania informacji uporządkowanej i wyselekcjonowanej zgodnie z wymaganiami odbiorcy. Model może być wykorzystywany na różnych poziomach szczegółowości informacji, postulowanej zarówno przez programistę, jak i użytkownika.

Przedstawiona koncepcja stnowi także spójny system, rozumiany jako taki, w którym brak wyraźnej różnicy między końcówką użytkownika a przyjętym sposobem programowania. Wprowadzenie inteligentnych baz danych było utrudnione do czasu rozwoju hipermediów, stacji roboczych i systemów eksperckich. Wskutek ewolucji tych technologii, inteligentne bazy danych mogą być obecnie użyte do wzbogacenia zasobu aktualnych danych oraz wykorzystywania informacji nawet przez mniej zamożnych użytkowników. Wszystkie bowiem zrelacjonowane w książce techniki mogą być wdrożone na komputerach osobistych.

* * *

Omawiana publikacja jest nowoczesnym podręcznikiem dla użytkowników baz danych, projektantów systemów eksperckich i entuzjastów hipertekstu. Technologia inteligentnych baz danych zapowiada innowacyjne zmiany w informatyce – od procesów badawczych i tworzenia nowych rozwiązań do metod przekazywania wiedzy. Książka może przyczynić się do powsta-

nia nowych generacji komputerów, gdyż wprowadza zunifikowany model pięciu technologii informacyjnych: bazy danych, programowania obiektowego, systemów eksperckich, hipermediów i zarządzania tekstem. Przedstawiono w niej wnikliwe wprowadzenie do hipertekstu w zakresie jego wykorzystania do reprezentacji wiedzy. Przedyskutowano programowanie obiektowe oraz zarządzanie tekstem, skupiając się głównie na konstrukcjach zdolnych do samomodyfikacji baz danych. W najbliższej przyszłości wraz z przyspieszeniem rozwoju technologicznego inteligentne bazy danych niewątpliwie zmienią nasz pogląd na informację w analogiczny sposób, jak rozpowszechnienie techniki komputerowej radykalnie przekształciło podejście do procesów obliczeniowych.

STEFAN JANUSZEWSKI
MARIA JANUSZEWSKA

Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej
Komitet Automatyki i Robotyki Polskiej Akademii Nauk Oddział Polskiej Akademii Nauk we Wrocławiu
organizują II Krajową Konferencję Naukową

INŻYNIERIA WIEDZY I SYSTEMY EKSPERTOWE

Wrocław 15–18 czerwca 1993 r.

Tematyka Konferencji:

- Problemy i metody sztucznej inteligencji
- Metodologiczne podstawy inżynierii wiedzy
- Metody statystyczne w inżynierii wiedzy i systemach ekspertowych
- Metody i komputeryzacja badań eksperymentalnych
- Podstawy budowy systemów ekspertowych
- Ekspertowe systemy sterowania, identyfikacji i rozpoznawania
- Zastosowania techniczne i biomedyczne

Komitet Programowy:

Prof. Zdzisław Bubnicki – przewodniczący
Doc. Adam Grzech – sekretarz naukowy
Prof. Michał Białko
Prof. Mirosław Dąbrowski
Prof. Zdzisław Hippe
Prof. Andrzej M. Janicki
Prof. Michał Kleiber
Prof. Juliusz L.. Kulikowski

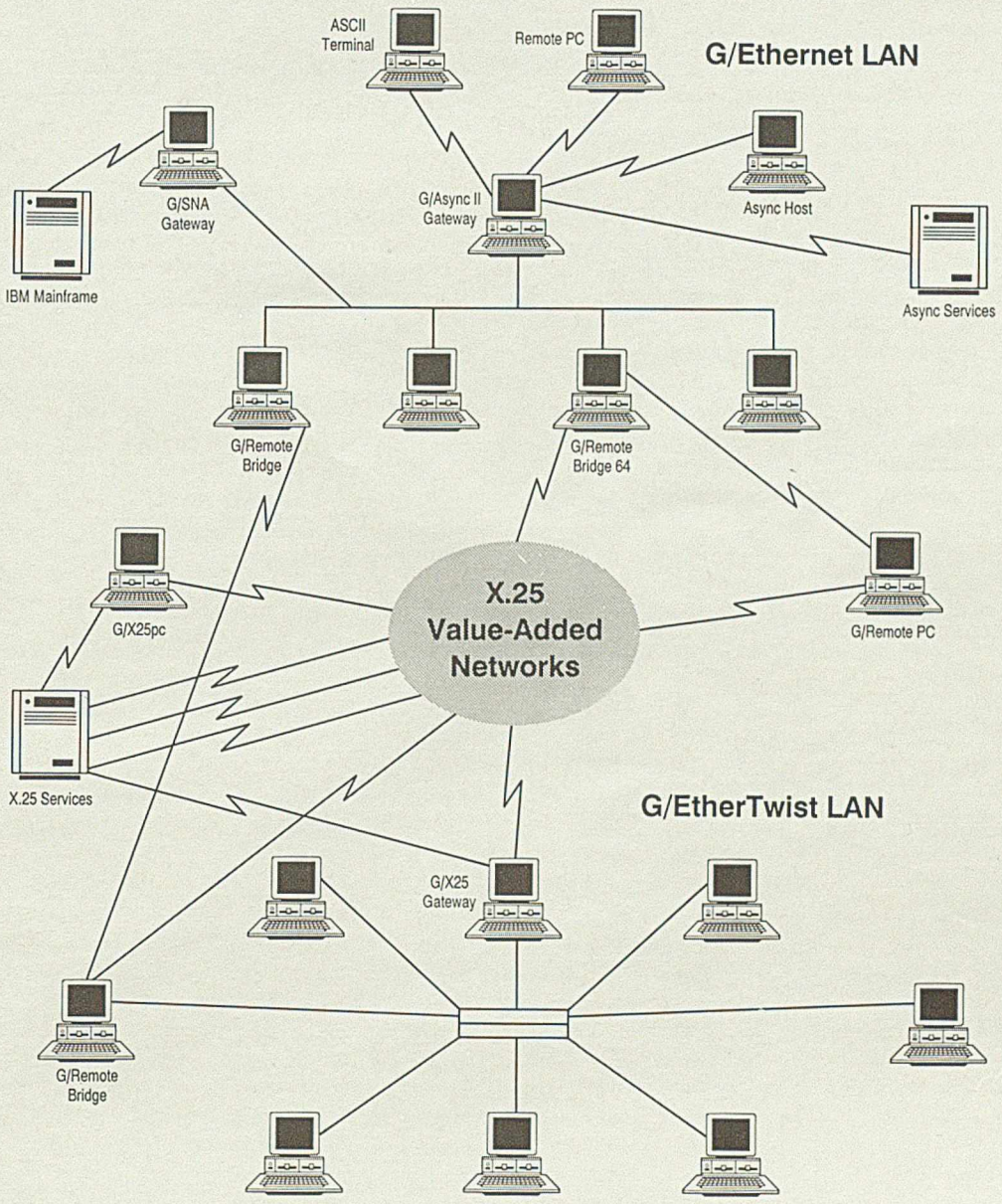
W czasie Konferencji odbędzie się sesja poświęcona prezentacji oryginalnego oprogramowania z zakresu inżynierii wiedzy i systemów ekspertowych.

Adres do korespondencji:
doc. Adam Grzech, Instytut Sterowania i Techniki Systemów, Politechnika Wrocławska, ul. Janiszewskiego 11/17, 50-370 Wrocław.

Berus T., Macukow B.: Od sieci neuronowych do neurokomputerów INFORMATYKA 1993, nr 1, s. 1 Charakterystyka najbardziej zaawansowanych modeli sieci neuronowych, wykorzystywanych do zastosowań komercyjnych.	**Berus T., Macukow B.: From neural networks to neurocomputers** INFORMATYKA 1993, No. 1, p. 1 Characteristics of the most advanced neural network models, used for commercial applications.	**Berus T., Macukow B.: Von Neuronennetzwerke bis Neurocomputer** INFORMATYKA 1993, Nr. 1, S. 1 Eine Charakteristik der fortgeschrittenen Modellen von Neuronennetzwerke, die zur kommerziellen Anwendungen ausgenützt wurden.
Wieczerzycki W.: Obiektowe bazy danych INFORMATYKA 1993, nr 1, s. 8 Geneza oraz zasady konstrukcji i działania obiektowych baz danych.	**Wieczerzycki W.: Object-oriented databases** INFORMATYKA 1993, No. 1, p. 8 Genesis as well as structure and operation principles of object-oriented databases.	**Wieczerzycki W.: Objektorientierte Datenbanken** INFORMATYKA 1993, Nr. 1, S. 8 Entstehungsgeschichte, sowie Konstruktions- und Betriebs grundlagen der objektorientierten Datenbanken.
Czajkowski G., Zieliński K.: STRAND88 – język współbieżnego programowania w logice INFORMATYKA 1993, nr 1, s. 13 Charakterystyka możliwości i przykłady stosowania języka współbieżnego programowania w logice STRAND88 oraz jego porównanie z językiem Prolog.	**Czajkowski G., Zieliński K.: STRAND88 – a language for parallel logic programming** INFORMATYKA 1993, No. 1, p. 13 Characteristics of possibilities and application examples of the STRAND88 language for parallel logic programming, as well as its comparison with the Prolog language.	**Czajkowski G., Zieliński K.: STRAND88 – eine Sprache für Parallelprogrammierung in Logik** INFORMATYKA 1993, Nr. 1, S. 13 Eine Charakteristik von Möglichkeiten und Anwendungsbeispiele der STRAND88-Sprache für Parallelprogrammierung in Logik, sowie ihr Vergleich mit der Prolog-Sprache.
Caban D.: Środki łączności w przemysłowych rozproszonych systemach sterowania INFORMATYKA 1993, nr 1, s. 19 Charakterystyka mediów transmisji, protokołów komunikacyjnych oraz metod dostępu, stosowanych w najbardziej znanych współczesnych rozproszonych przemysłowych systemach sterowania.	**Caban D.: Communication media in distributed industrial control systems** INFORMATYKA 1993, No. 1, p. 19 Characteristics of communication media, protocols and access methods, applied in the most famous contemporary distributed industrial systems.	**Caban D.: Kommunikationsmittel in verteilten Industriesteuersystemen** INFORMATYKA 1993, Nr. 1, S. 19 Eine Charakteristik von Kommunikationsmedien, Protokollen und Zugriffsmethoden, die in bekanntesten zeitgenössischen verteilten Industriesteuersystemen angewendet wurden.

Produkty sieciowe LAN i WAN

Gateway communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:

PC MAGAZINE, LAN MAGAZINE, INFO WORLD

OFERUJE AUTORYZOWANY DYSTRYBUTOR:

MIKROB

P.W.P.T. MIKROB SP. Z O.O.

20-346 Lublin, ul. Długa 5

Tel. (0-81) 420-61, faks 415-43, teleks 643776 MIKRO PL

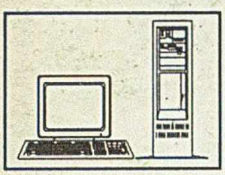
PC MAGAZINE EDITORS' CHOICE January 31, 1989 G-Ethernet AT

PC MAGAZINE EDITORS' CHOICE October 16, 1990 G-EtherTwist

0/15/91

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.

JUNISOFTEX®



Przedsiębiorstwo Produkcyjno-Usługowe Innowacji Technicznych Biuro Techniczno-Handlowe 44-100 Gliwice ul. Konstytucj 11.
tel. 31-57-00, 31-51-31, 31-75-10, fax 31-75-10, centr. 31-90-81 do 88, telex 036233
Zespół wdrożeń i analiz systemów komputerowych wewn. 15 ,25, 282, Zespół inżynierii oprogramowania wewn. 14, 20, 245,
Zespół d/s serwisu systemów Novell i sprzętu komputerowego wewn. 16, 245, Dział Handlowy wewn. 18, 28, 201,
Szkoła Komputerowa wewn. 282, Sekretariat wewn. 250, Prezes wewn. 13, 272, Dyrektor wewn. 12, 272, Główny Księgowy wewn. 256,
** Konto: PKO I/O Gliwice 27661-225298-136 **

OFERUJEMY

Kompleksową komputeryzację przedsiębiorstw obejmująca dostawę i wdrożenie zintegrowanych systemów komputerowych takich jak system:

finansowo-kosztowy
(automatyczne rozliczanie kosztów)
+
gospodarka materiałowa
+
środki trwałe
+
wyposażenie
+
obrót towarowy fakturowanie sprzedaży
+
kadry-płace
+
techniczne przygotowanie produkcji

Sprzęt komputerowy w tym:

Komputery
Monitory
Dyski twarde
Drukarki
(mozaikowe i laserowe)
osprzęt sieciowy
(karty sieciowe, kable zwykłe i światłowodowe)

Systemy i oprogramowanie licencjonowane

DOS
Novell
Btrieve
SQL Base
Windows

Szkolenia komputerowe

podstawy obsługi komputerów
arkusze kalkulacyjne (Qpro, Lotus, EXCEL)
edytory tekstów (WordPerfect, TAG, ChiWriter)
bazy danych (dbase IV, Clipper, Paradox)
systemy operacyjne (DOS, Novell)

SOFTWARE SUPPORT FOR DOS, NOVELL 2.2, 3.11, 4.0, SQL BASE firmy GUPTA

*

OPROGRAMOWANIE DLA PRZEDSIĘBIORSTW

oferujemy własnej produkcji zakładowy system informatyczny PHU-Perfect(N) obejmujący swoim zakresem:
system finansowo-kosztowy umożliwiający automatyczne rozliczanie kosztów, atomatyczną dekretację, analizę rozrachunków
system obrotu materiałami, wyrobami, towarami (kody paskowe) w tym: stany magazynowe, ewidencje zamówień, fakturowanie sprzedaży (VAT) z automatyczną dekretacją rozdzielników kosztów i sprzedaży do systemu finansowo-kosztowego,
system ewidencji majątku trwałego i wyposażenia
system kadrowo-płacowy (SQL) z automatyczną dekretacją rozdzielników płacowych do systemu finansowo-kosztowego
system technicznego przygotowania produkcji (uk. 1993 r)
(oprogramowanie w systemie SQL BASE w przygotowaniu)

*

OPROGRAMOWANIE DLA URZĘDÓW MIAST I GMIN

oferujemy własnej produkcji system finansowo-podatkowy UMG-Perfect(N) obejmujący swoim zakresem:
system finansowo-księgowy,
systemy podatkowe obejmujące: ustalanie wymiarów, naliczanie i kontrolę płatności należności podatkowych dla dowolnych podatników i/lub płatników podatków na ich kontach i kontach dochodów ze swobodną korektą dowolnych informacji w dowolnym okresie a w tym:
ewidencja zapłat z wykorzystaniem kodów paskowych, emisja wezwań do zapłaty, naliczanie odsetek.
system ewidencji majątku

*

Zapraszamy do współpracy w zakresie wdrożeń naszych systemów partnerów, - firmy informatyczne z całego kraju !!!